UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

TELEMÁTICA

PROYECTO FINAL DE CARRERA

Desarrollo de un sistema de posicionamiento de teléfonos móviles en J2ME

Autor: Francisco José Sánchez Bermúdez
Tutor: Mario Muñoz Organero

Julio de 2010

Agradecimientos

He considerado importante dejar para el final de todo el proyecto los agradecimientos, porque no se trata de un proceso científico, ni analítico, se trata de hablar arropado con un manto de sensaciones próximas al paroxismo. No estoy escribiendo un apartado de mi proyecto, estoy contando y agradeciendo por todo un importante apartado de mi vida. Por ello, mi forma de expresión y pensamiento en condición de ingeniero debía quedar a un lado para "razonar" con el corazón.

No puedo por más que gritar por dentro un sonoro y sincero GRACIAS que resonará en mí para el resto de mi existencia. El torbellino de emociones que se agolpan por expresarse deja tras de sí dos personas inmensas sin las cuales no soy nada, mis tablas de salvación, mis esperanzas en la adversidad, mis guías de este laberíntico mundo, MIS PADRES. Es fácil jugar tamaña partida con tan buenas cartas, la suerte puede esperar siempre que ellos permanezcan ubérrimos en mi alma. Con una subversión aplastante aportan a mi diccionario palabras tales como constancia, superación, esfuerzo, amor o paciencia. Por su eterna comprensión, por su irreverente forma de ver las cosas, por su templanza en ocasiones y su exigencia en otras, simplemente POR ELLOS estoy redactando las notas finales de este proyecto y esta aventura.

No perdonaría mi recuerdo olvidarse de nadie, por tanto un hueco preferencial en él por lo que suponen en mi vida está reservado para mi tía y mi abuela, por su amor infinito y su incondicional confianza que fortifican mi fuerza de voluntad. Un pasado florido de grandes recuerdos afianza una vida de buenos momentos. Muchísimas gracias por vuestras lecciones de vida y titánico apoyo durante tantos años.

Llegados a este punto, voy a permitirme la licencia de hablar sólo para ella. Va por ti, por lo grande que has sido, por todo lo que nos has dejado, porque has sabido, incluso, sobreponerte a hechos ineludibles del destino como sólo las grandes señoras pueden hacer. Conocido es que somos un juguete en manos del caprichoso Dios Chronos, pero no es menos cierto que en medio de este trajín hay momentos, o mejor dicho, hay personas capaces de, en su grandeza, menospreciar a este Dios, tan incorpóreo como engreído y robarle su bien más preciado. Semejantes personas son imborrables en nuestra memoria, en nuestra alma y es por esto por lo que quiero que sepas que, aunque seguro que estarás de viaje y lo leerás más tarde, va por ti.

El recuerdo encamina mis pasos hacía una gran persona que vivió en un constante reparto de amor y fraternidad. Debería estar aquí para cumplir uno de sus anhelos, pero bien sé que allá arriba se removerá de felicidad cuando vea mi carrera concluida.

En esta serpenteante carretera de montaña, en ocasiones hay seres que ofrecen de manera altruista toda el agua que puedan tener para la difícil subida, eliminando incluso su ración correspondiente. Cuando alguien tiene la inmensa suerte de que le sucede este extraño hecho debe apreciarlo, mostrarse ampliamente congratulado y dar gracias hasta la extenuación por el apoyo, la ayuda, la comprensión, la paciencia, el amor y el desinterés constante. En mi caso, el inefable y querido ser de montaña, responde al nombre de Eva, máxima expresión del "todo tiene quien todo da"; Muchísimas gracias.

Un papel principal en mi vida es representado por mi círculo más cercano de gente, mis amigos, una segunda familia. Algunos los he conocido en la carrera o mediante ella y otros llevan conmigo y mis peculiaridades toda una vida. Por todos los momentos vividos, por los que quedan por vivir y por lo que ha supuesto una sólida amistad en esta carrera, os doy las gracias.

No es usual este tipo de agradecimientos, quizá no sea de recibo, pero no puedo permitir olvidar a un gran amigo. Su lealtad, fidelidad y compañía en todos y cada uno de los momentos pasados durante esta carrera merecen la pena ser recordados. Se dice que el tiempo todo lo cura, pero no es capaz de eliminar recuerdos incandescentes de la memoria, por ello este es tu merecido homenaje.

Compañeros de fatigas deben ocupar su lugar en este apartado ya que han sido un apoyo, vía de escape y ayuda durante todo este largo proceso de "ingenierización". He optado por no personalizar demasiado a lo largo de estas líneas, por tanto espero que todos ellos se sientan congraciados ya que se lo merecen por todo lo que me han aportado.

Por su inestimable profesionalidad, su constante predisposición para ayudar, así como su coordinación y total apoyo para cualquier necesidad surgida durante todo el desarrollo del proyecto, mi tutor, Mario, merece no menos que un GRACIAS mayúsculo. En estos tiempos de endémicas prisas, no abundan personas de estas características y por ello le brindo mi gratitud.

Todos y cada uno de los aquí mencionados han hecho posible que alcance esta gran meta, por ello mi más sincero y emocionado agradecimiento.

Resumen

Los actuales dispositivos móviles incluyen un sinfín de funcionalidades. Esta inercia nos lleva a que constantemente encontremos, implementadas en los terminales, nuevas actualizaciones tecnológicas. De entre todas las tecnologías que incorporan hay dos que actualmente, por diversos motivos, son las más populares. Por un lado GPS es un sistema de posicionamiento que está en constante evolución, lo que supone que los usuarios exijan cada vez más soluciones de localización. Hoy en día muchos dispositivos incluyen este sistema que ofrece información acerca de la posición del usuario utilizando varios satélites. Por otra parte la tecnología inalámbrica más extendida y utilizada en dispositivos móviles en la actualidad no es otra que Bluetooth, cuyo desarrollo se inició en 1994 y sus creadores y demás empresas asociadas continúan lanzando al mercado nuevas especificaciones.

A lo largo de este proyecto la intención será “unir” estas dos tecnologías en auge en un solo sistema. El objetivo del proyecto es poder ofrecer al usuario un sistema de localización similar a GPS utilizando para ello, únicamente, el intercambio de información mediante Bluetooth. Toda vez que utilizamos esta tecnología inalámbrica nuestro sistema estará limitado por el alcance de cobertura que nos pueda ofrecer.

Mediante la descripción detalla, tanto de bajo como alto nivel, de nuestro sistema se podrá comprender su modo de funcionamiento simple y efectivo. Además se dispondrá de varios entornos reales de pruebas mediante los cuales se podrá refrendar la nueva funcionalidad que aporta.

El sistema planteado permite una comunicación vía Bluetooth entre el usuario que requiere información de localización y los dispositivos que se la proporcionan.

Abstract

Current mobile devices include a host of functionalities. This inertia leads us to constantly meet new technology updates implemented in the terminals. Among all the technologies there are two that they are the most popular for various reasons. On the one hand GPS is a positioning system that is constantly evolving, which means that users increasingly require location solutions. Nowadays, many devices include this system that provides information about the user's position using multiple satellites. Moreover, the most widespread wireless technology and mobile devices used today is none other than Bluetooth, whose development began in 1994 and its creators and other associated companies continue to launch new market specifications.

Throughout this project the intention is to "link" these two technologies on the rise in a single system. The project's goal is to offer users a positioning system similar to GPS using only information exchange through Bluetooth. Whenever we use this wireless technology our system is limited by the scope of coverage that we can offer.

Through detailed description of our system is able to understand its mode of operation simple and effective. In addition, we have many test environments by which we will endorse the new functionality it provides.

The proposed system allows communication via Bluetooth between the user that requires location information and the devices that provide it.

Índice general

Índice de Figuras

Índice de Tablas

1 INTRODUCCIÓN Y OBJETIVOS

1.1 MOTIVACIÓN

El incesante avance en el desarrollo de los terminales móviles así como las exigencias y necesidades de los usuarios han llevado a que un dispositivo funcione como cámara fotográfica, modem, radio o receptor WiFi entre otras muchas utilidades. Sin embargo para que un terminal de gama media-alta pueda tener hueco en el mercado y codearse con los demás competidores es requisito prácticamente indispensable que incorpore GPS o, al estilo de los últimos lanzamientos, GPS asistido (A-GPS). A su vez, hoy en día encontramos una ingente cantidad de dispositivos externos, incorporados recientemente al mercado, que se comunican con nuestro terminal mediante la tecnología inalámbrica más usada en telefonía móvil, Bluetooth. Esta nueva era de dispositivos como pueden ser auriculares, manos libres, altavoces y un largo etcétera están, en la actualidad, completamente integrados en nuestras vidas logrando, de manera constante, incrementos en sus ventas. Como consecuencia, Bluetooth está presente en cualquier tipo de terminal, sea de gama baja como Premium, y ha relegado al ostracismo a otra tecnología inalámbrica, infrarrojos, muy valorada hace varios años para el intercambio de información entre terminales.

El desarrollo de nuestro sistema tiene como elementos principales fusionar la utilización de la tecnología inalámbrica y el posicionamiento de dispositivos móviles dentro de un espacio de cobertura determinado por dicha tecnología.

Analizando diferentes características de cada uno de los tipos de especificaciones de tecnologías sin cables, hemos observado que para el sistema de posicionamiento que queremos implementar Bluetooh es la tecnología más apropiada. Está en continuo desarrollo y es ampliamente utilizada en diversos ámbitos, siendo el más común el de los dispositivos móviles. Nuestro sistema no podría utilizar infrarrojos debido a la escasa cobertura. Este problema sí lo solventan otras tecnologías como puede ser WiFi, sin embargo utilizamos Bluetooth por su simplicidad y bajo coste frente a ésta. Por otra parte, otra especificación inalámbrica como puede ser ZigBee no está pensada para dispositivos móviles.

Como primera aproximación, ignorando datos de bajo nivel, nuestro programa será capaz de ofrecer la localización de un dispositivo móvil dentro de un área limitada tomando un punto de referencia inicial que nosotros estimemos conveniente. Esto, como se ha especificado anteriormente, se consigue mediante el envío de información por Bluetooth.

Es importante remarcar que no se han utilizado tecnologías reales de posicionamiento como pueda ser GPS. En un primer momento se consideró viable utilizar la librería de localización para J2ME, JSR 179, que nos ofrece funcionalidad relacionada con la posición geográfica de nuestro dispositivo. Esta opción se desechó por varios motivos determinantes, uno de los cuales fue que las operadoras ofrecen servicios de posicionamiento previo pago. Como resultado de esta decisión, nuestro programa es capaz de ofrecer información de posicionamiento de forma rápida, segura, barata, fiable (los errores de posicionamiento están dentro de valores aceptables), eficiente y utilizando una tecnología inalámbrica actualmente presente en todos los terminales con un crecimiento excepcional.

Para conseguir esto, como cabe esperar, se dispondrá de un sistema cliente-servidor, de forma que la aplicación cliente peticione la información de localización necesaria cuando el usuario así lo requiera. Estas peticiones serán recibidas por un número determinado de balizas o servidores (dependiendo de las que se encuentren dentro de la zona de cobertura Bluetooth del dispositivo) que proporcionarán su información de localización almacenada. La aplicación cliente, una vez recibida la información de las balizas, cruzará los datos y, mediante la ejecución de un algoritmo, obtendrá su posición actual.

Se han analizado diferentes algoritmos a lo largo del desarrollo de nuestro sistema por lo que más adelante los veremos con mayor detalle. Con cada tipo de algoritmo utilizado obtendremos un error medio diferente con respecto a la ubicación real de nuestro dispositivo móvil.

Una descripción sencilla de lo que es nuestro sistema viene dada en la figura1.1. Como puede observarse, consta de una clase inicial (Midlet) mediante la cual podemos elegir el modo de funcionamiento de la aplicación, cliente o servidor. Dependiendo de la elección del usuario se creará uno u otro objeto con sus valores correspondientes para el manejo de Bluetooth en el terminal.

De este diagrama de clases simple extraemos la sencillez de ejecución de nuestro programa, con lo que somos capaces de obtener un mayor rendimiento, velocidad, ahorro de baterías, memoria y recursos del procesador frente a otras aplicaciones que utilicen herramientas de posicionamiento reales.

A lo largo de esta memoria se dará una explicación detallada del funcionamiento de nuestro sistema así como de sus posibles usos, mejoras y pruebas realizadas sobre la misma.

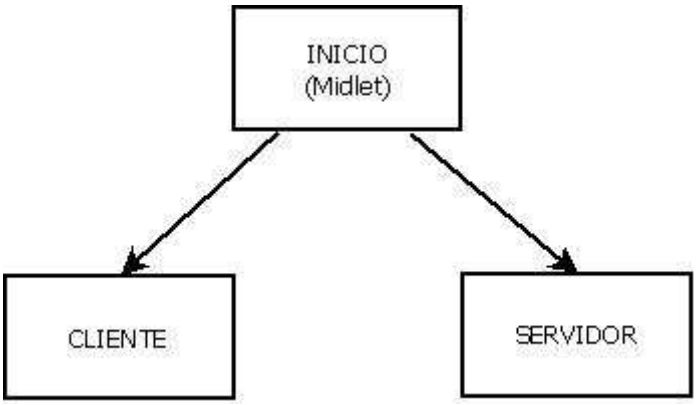

Figura 1.1: Esquema genérico del sistema

Adjunto a esta aplicación se ha realizado un pequeño programa de testeo de la misma. Utilizando los diferentes algoritmos estudiados para el caso que nos ocupa, reportará unos errores medios característicos de cada uno de ellos tomando como ubicación del dispositivo un punto generado aleatoriamente. Estos errores han sido almacenados y se detallarán más adelante para comprobar la fiabilidad del uso de cada uno de los algoritmos empleados para obtener la posición del dispositivo móvil. A su vez, el número de balizas o servidores a utilizar podrá ser insertado por el usuario o generado aleatoriamente. El posicionamiento de los mismos también se efectuará de manera aleatoria.

1.2 CONTENIDO DE LA MEMORIA

En este apartado se dará una breve descripción de cada uno de los capítulos y apéndices de los que consta esta memoria.

1. Capítulo 2: Estado actual de las tecnologías utilizadas en nuestro sistema así como lanzamientos de futuras versiones. Se pondrá especial atención tanto en la tecnología inalámbrica Bluetooth como en la librería utilizada en J2ME para manejarla, JSR82.

2. Capítulo 3: Amplia descripción sobre las características y diseño del sistema implementado.

3. Capítulo 4: Análisis y descripción del código y funcionamiento de las aplicaciones realizadas.

4. Capítulo 5: Descripción de las pruebas de simulación y los datos reportados por las mismas, así como de la aplicación creada para tal fin.

5. Capítulo 6: Apartado descriptivo de todas las pruebas realizadas con terminales móviles en los distintos escenarios planteados.

6. Capítulo 7: Desglose del apartado monetario asociado a la consecución de este proyecto.

7. Capítulo 8: Conclusiones obtenidas una vez implementado todo el sistema y realizadas todas las pruebas posibles.

8. Capítulo 9: Trabajos futuros que puedan modificar o mejorar el sistema implementado.

9. Capítulo 10: Diagrama de Gantt con todas las tareas realizadas para la consecución de este proyecto y sus tiempos de ejecución.

10. Apéndice A: Distribución de las balizas en cada uno de los escenarios pensados para las pruebas de campo.

11. Apéndice B: Nombres y valores de los UUIDs de Bluetooth de SDP.

12. Apéndice C: Manual de utilización del sistema real, para su correcto uso y funcionamiento en dispositivos móviles.

13. Apéndice D: Contenido del CD adjunto, así como utilización de las aplicaciones y descarga y manejo de entornos de desarrollo correspondientes.

1.3 CONVENCIONES DE FORMATO

A lo largo de esta memoria se utilizarán los formatos que aparecen en la tabla para tratar distintos aspectos. Texto, tablas y código contarán con un estilo propio.

FORMATO	DESCRIPCIÓN
Times New Roman	Usado para texto
Arial	Usado para tablas
`Courier New`	`Usado para código`

2 ESTADO DEL ARTE

En este capítulo, para conocer más extensamente las especificaciones del sistema, daremos una breve descripción de la tecnología Bluetooth, ampliamente conocida, y de la librería JSR 82, utilizada en J2ME para el manejo de aquélla.

Un vertiginoso avance técnico ha llevado a que, desde su aparición en 1994, Bluetooth presente diversas versiones. Las últimas fueron desarrolladas en 2009, a mediados la versión 3.0 y a finales la 4.0. Si bien, ni en los más modernos dispositivos móviles se encuentran instaladas aún. Actualmente la mayoría de dispositivos cuentan con la versión 2.0 con A2DP, adoptada en 2004, llegando algunos de clase superior a incorporar la 2.1+EDR[1] . Nosotros trabajaremos y nos referiremos siempre a V2.0, siendo A2DP un perfil que define como propagar streams de audio, mono o estéreo, entre dispositivos a través de Bluetooth. La versión 2.1 incorpora mejoras en los emparejamientos de los dispositivos, consumo de baterías (5 veces menos) y seguridad.

Como nota final al respecto de las versiones, la última, desarrollada en diciembre del 2009 (V.4.0), tendrá un mejor comportamiento frente al consumo, bajo coste, menor latencia y una fuerte encriptación y autenticación de los paquetes enviados entre otras muchas novedades[2]. El SIG ha anunciado que podría llegar al mercado a finales del segundo trimestre de este año, posiblemente entre Junio y Julio.

2.1 CARACTERÍSTICAS BLUETOOTH

Dado que esta tecnología es ya conocida por todos los usuarios, nos adentraremos más a fondo sólo en aquellas características que son importantes para el desarrollo de nuestra aplicación.

Nuestro sistema de posicionamiento está pensado para que sea efectivo dentro de un determinado entorno y con un limitado radio de funcionamiento. Este límite funcional se debe a la cobertura máxima que puede ofrecer un dispositivo Bluetooth. Como especificación técnica observamos que la clase 2 puede tener una cobertura óptima de 10 metros.[3]

Con este dato podemos asegurar que nuestro dispositivo móvil detectará todos los servidores de información de localización en un radio mínimo de 10 metros en zonas diáfanas.

Clase	Potencia máxima(dbm)	Cobertura óptima(aprox.)
1	20	100
2	4	10
3	0	1

Tabla 2.1: Rangos de coberturas Bluetooth óptimos

Este dato en cuanto a su cobertura es ampliamente aceptado, si bien, en una de las diferentes pruebas realizadas para la consecución de este proyecto se han observado medidas máximas que son importantes recalcar. Dichas medidas se han realizado tanto en zonas diáfanas como en otras con algún tipo de obstáculo. Las pruebas están realizadas en un entorno de interiores, colocando los dispositivos a lo largo de un pasillo lo suficientemente extenso para comprobar los cálculos.

	Cobertura máxima
Sin obstáculo	19m
Con obstáculo	8-9m

Tabla 2.2: Coberturas máximas

Sobre estos valores obtenidos en pruebas se ha analizado la influencia de otra variable. Posicionando un dispositivo en una esquina del pasillo, lograremos un mayor espacio de cobertura cuanto más nos acerquemos a la pared contraria. Este hecho se verá más ampliado en el capítulo de pruebas de campo.

Otra de sus características es la capacidad de establecer comunicaciones full duplex y punto a multipunto, lo que nos ofrece un adelanto en lo que a la comunicación entre cliente y balizas se refiere.

En cuanto a las velocidades de transferencia, contamos con hasta 3Mbps en la especificación utilizada como muestra la tabla 2.3. Esta tasa de transmisión de datos es más que suficiente para nuestro sistema ya que la información intercambiada no tiene mucho peso. Por tanto la información de posicionamiento se transferirá de manera rápida. Con esta gran velocidad de transferencia se pueden plantear mejoras en nuestra aplicación en las que haya un intercambio de mayor cantidad de datos como podremos ver en el capítulo 8.

Versión	Velocidad
1.2	1 Mbps
2.0	3 Mbps
3.0	>24Mbps

Tabla 2.3: Velocidades de transferencia

Por otra parte, en el caso que nos ocupa, la seguridad no es considerada un aspecto primordial debido a que los datos de localización del dispositivo, en la mayoría de los entornos, no serán de vital importancia. Si bien, manipular estos datos puede dar lugar a grandes confusiones por parte del cliente. Por ello se han estimado suficientes los mecanismos de seguridad de Bluetooth y no se considera necesario implementar algún otro para esta primera versión de nuestra aplicación. Estos mecanismos son un cifrado de 128 bits y autenticación por pin. A su vez, para evitar interferencias, incluye un mecanismo de saltos de frecuencia que ofrece una gran funcionalidad incluso en entornos de altas interferencias.

A modo de ejemplo en este apartado, sin entrar en descripciones, se muestra un dispositivo que cumple todas las características mencionadas anteriormente. Se trata del transductor LMX5252. Este chip, distribuido por National, es uno de los más pequeños en la industria, implementado en 0,18 micras[4].

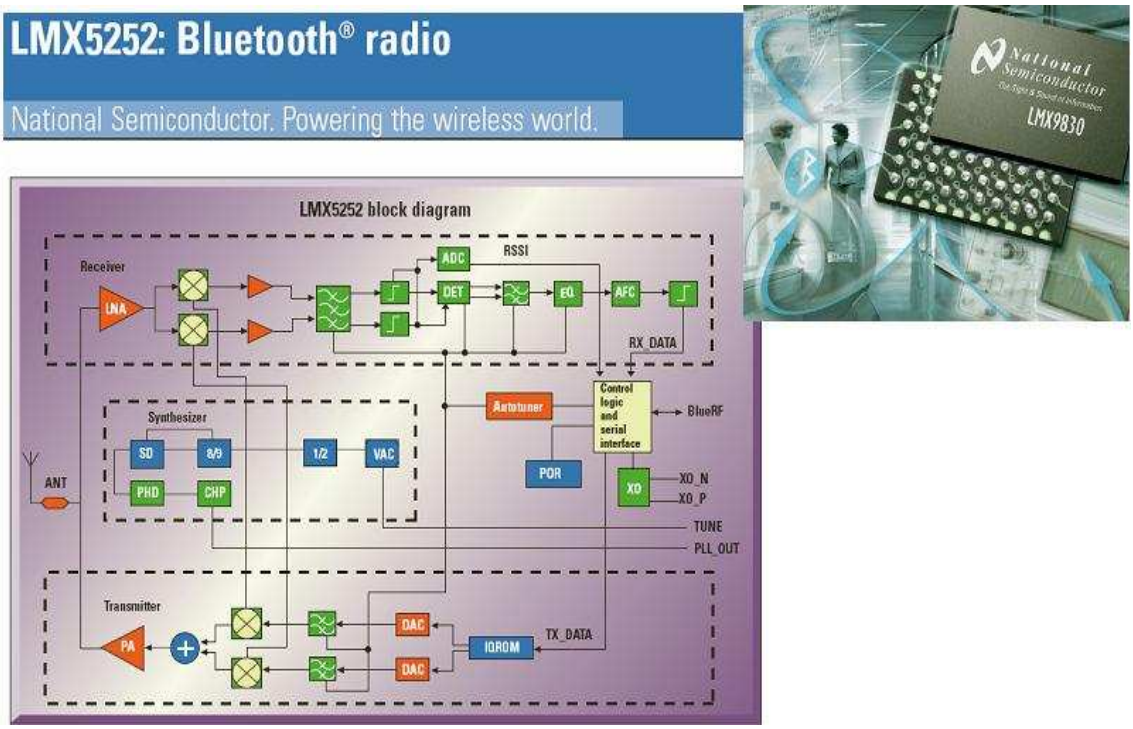

Figura 2.1: Transductor LMX5252

2.1.1 PROTOCOLOS BLUETOOTH

Una parte importante que ha de saberse en lo que a características Bluetooth se refiere es la pila de protocolos. Como punto de partida, vamos a tomar la figura 2.2 que muestra esquemáticamente la disposición de todos los protocolos que intervienen en la transmisión de datos mediante Bluetooth. Únicamente describiremos aquellos que consideramos más importantes, así como los más característicos de nuestro sistema.

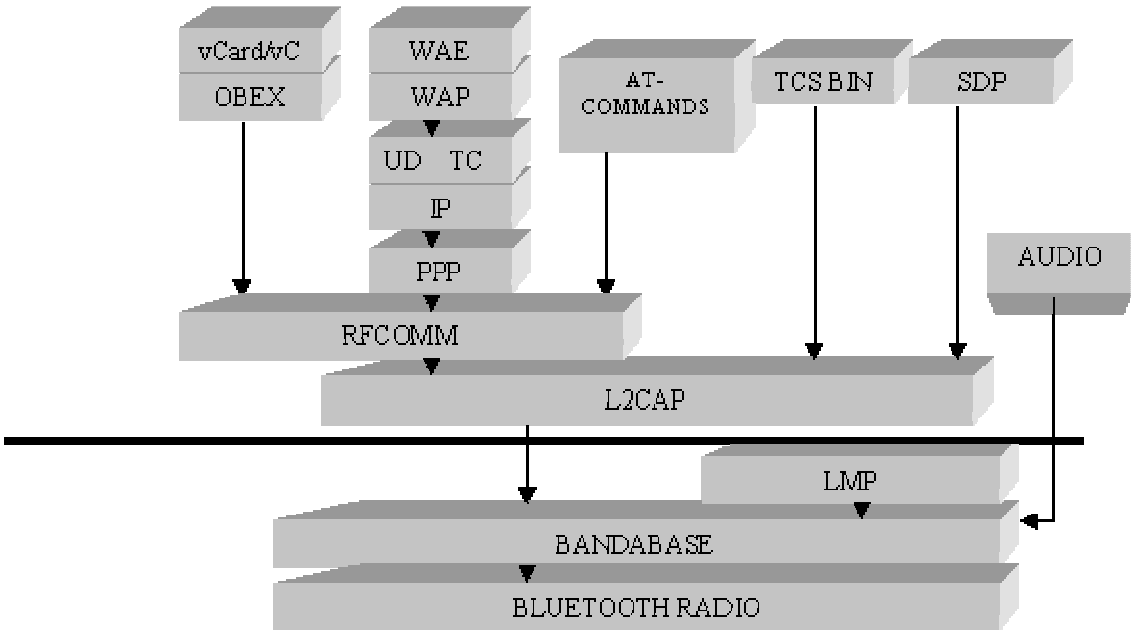

Figura 2.2: Pila de protocolos Bluetooth

L2CAP

Proporciona a los protocolos de niveles superiores multiplexación, segmentación y reensamblado de paquetes y gestión de la calidad de servicio. Puede actuar en modo orientado y no orientado a conexión y se ubica en la capa de enlace[5].

RFCOMM

Es un protocolo muy importante en nuestro sistema. Se construye sobre L2CAP y emula puertos serie RS-232. Está basado en el estándar ETSI TS 07.10[6]. El puerto serie de Bluetooth está basado en este protocolo y está estrechamente alineado con el perfil SPP (Serial Port Profile). SPP facilita comunicaciones entre dispositivos Bluetooth ya que provee al protocolo RFCOMM de un interfaz basado en streams. Estas son algunas de sus características más importantes[7]:

1. Más de 60 conexiones simultáneas
2. El número de conexiones depende de la implementación
3. Dos dispositivos sólo pueden compartir una sesión RFCOMM a la vez
4. Un solo dispositivos puede tener hasta 30 servicios RFCOMM activos

Para las conexiones en nuestro sistema, implementadas mediante JSR82, utilizaremos este protocolo de transporte.

OBEX

Protocolo desarrollado por IrDA con una funcionalidad similar a HTTP que puede ser usado por diferentes medios de transmisión (Infrarrojo, Bluetooth…) para el intercambio de datos.

SDP

Al igual que RFCOMM, SDP (Service Discovery Protocol), juega un papel importante en nuestro sistema. Este protocolo permite a las aplicaciones descubrir cuáles son los servicios disponibles en un dispositivo Bluetooth (en nuestro caso las balizas o servidores). Además, determina las características de esos servicios.

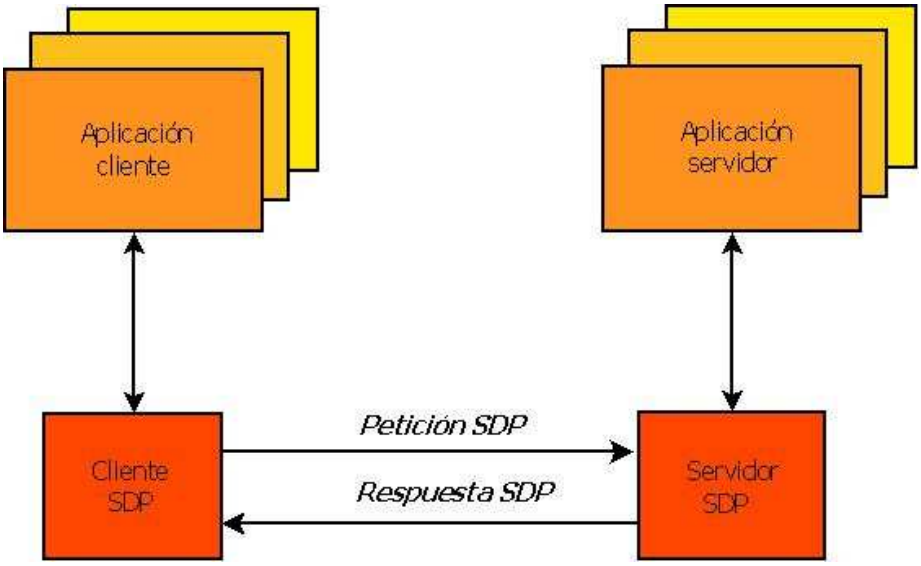

Figura 2. 3: Funcionamiento SDP

SDP sigue un modelo petición-respuesta y, como se puede observar, lo usa tanto el cliente como el servidor para poder obtener la información acerca de cualquier servicio disponible.

Para finalizar este resumen de protocolos, en la figura 2.4 se muestra, como complemento, el formato de paquetes enviados. La cabecera BT es de 54 bits.

Esta cabecera contiene importante información de control (tipo de paquete, control de flujo, dirección del dispositivo…). Por otra parte utiliza 72 bits para el código de acceso que se deriva de la entidad maestra.

Figura 2.4: Formato de paquete Bluetooth

2.1.2 PERFILES BLUETOOTH

Los perfiles Bluetooth tienen una importancia mayúscula en el uso de esta tecnología. Especifican el uso que puede hacerse de un dispositivo que implementa Bluetooth. Son las guías que detallan los procedimientos mediante los cuales dos dispositivos son capaces de comunicarse, es decir, determinan las pautas a seguir para establecer, mantener y cerrar conexiones. Dicho lo cual, como cabe esperar, para cualquier dispositivo que pretenda utilizar esta tecnología inalámbrica, es requisito indispensable que disponga de, al menos, un perfil Bluetooth.

Estas "guías de funcionamiento" se encuentran estrechamente ligadas con la pila de protocolos Bluetooth. Esto se debe a que, para cada protocolo, definen opciones obligatorias para un determinado perfil y rangos de parámetros necesarios.

Como puede intuirse los perfiles, vistos como plantillas genéricas, fueron creados para establecer un consenso entre los fabricantes. De esta forma los desarrolladores pueden crear cualquier aplicación compatible con otros dispositivos, siempre y cuando se ajusten a estas plantillas preestablecidas.

Para aportar una información detallada y completa incluimos algunos de los perfiles Bluetooth más utilizados en la tabla 2.4, incidiendo sobre los más importantes para nuestro sistema más adelante.

A2DP	Advanced Audio Distribution Profile
AVRCP	Audio Video Remote Control Profile
BIP	Basic Imaging Profile
BPP	Basic Printing Profile
CIP	Common ISDN Access Profile
CTP	Cordless Telephony Profile
DNP	Dial-up Networking Profile
ESDP	Extended Service Discovery Profile
FP	Fax Profile
FTP	File Transfer Profile
GAP	**Generic Access Profile**
GAVDP	Generic Audio Video Distribution Profile
GOEP	Generic Object Exchange Profile

HCRP	Hardcopy Cable Replacement Profile
HFR	Hands-Free Profile
HID	Human Interface Device Profile
HS	Headset Profile
IP	Intercom Profile
LAP	LAN (Local Area Network) Access Profile
OPP	Object Push Profile
PAN	Personal Area Networking
SAP	SIM Access Profile
SDAP	Service Discovery Application Profile
SP	Synchronization Profile
SPP	**Serial Port Profile**

Tabla 2.4: Perfiles Bluetooth

Cada uno de los perfiles mostrados debe incluir, de forma obligatoria, información acerca de la dependencia con otros perfiles, formatos de la interfaz de usuario más convenientes, así como características concretas de la pila de protocolos Bluetooth utilizada. A su vez, opcionalmente, puede incluir un breve resumen de los servicios requeridos.

GAP

El perfil de acceso genérico constituye la base para la totalidad de los perfiles restantes. Todos ellos dependen de éste, por lo que puede considerarse como el perfil más importante. Esencialmente define los pasos a seguir para crear un enlace entre dos dispositivos.

Por tanto para que dos dispositivos puedan tener cierta compatibilidad deben ajustarse, como mínimo, al perfil GAP, independientemente de sus fabricantes o aplicaciones disponibles.

SPP

Ante todo, este perfil es el que utilizaremos para intercambiar cualquier tipo de información en nuestro sistema. Se trata de un perfil muy importante ya que también sirve de base para otros muchos, como ocurría con GAP.

El perfil de puerto serie define los requisitos necesarios para que dos dispositivos puedan emular una conexión de puerto serie RS-232 entre ellos, usando como protocolo RFCOMM[8].

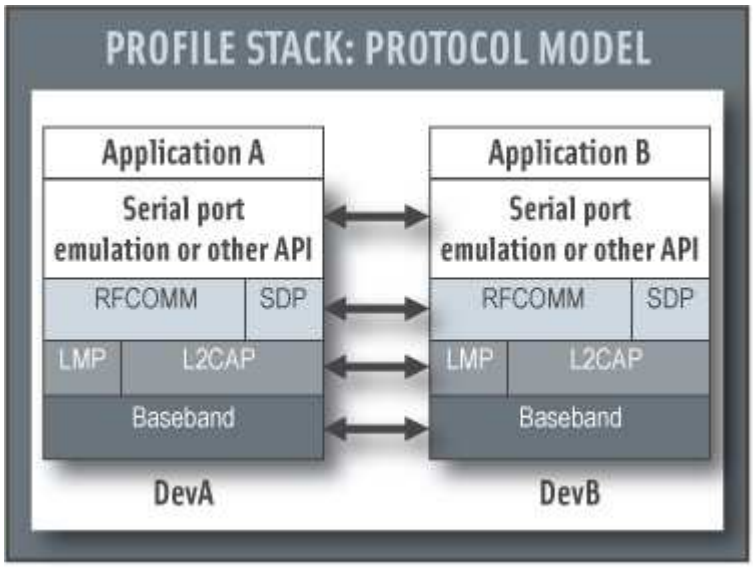

Figura 2.5: Funcionamiento SPP

El dispositivo A toma la iniciativa para crear la conexión, mientras que el B espera a que algún dispositivo la haya tomado. Ambos se conectan de forma inalámbrica, emulando una conexión por cable.

2.2 CARACTERÍSTICAS JSR82

A lo largo de este apartado veremos una pequeña introducción de las propiedades y clases que podemos utilizar de esta API de J2ME. No entraremos a detallar la funcionalidad de las clases que aporta ya que esto se hará, concienzudamente, en los capítulos de descripciones posteriores.

En la actualidad, contamos con la especificación 1.1.1 lanzada el 29 de Julio del 2008 y que cuenta con ciertas mejoras en cuanto a errores respecto a sus versiones precedentes. Este último lanzamiento ha sido implementado por un nutrido grupo de compañías punteras en el mundo de las comunicaciones, como pueden ser IBM, Nokia, Symbian, Sun, Motorola o Ericsson, que inició en el año 1994 el desarrollo de Bluetooth.

Las funciones ofrecidas por esta librería las podemos clasificar en tres categorías; *descubrimiento* de dispositivos o servicios y registro de los mismos, *comunicación* entre dispositivos y aplicaciones y *gestión de dispositivos* para controlar las comunicaciones establecidas.[9].

Figura 2.6: Funciones JSR82

Debido a esta sencilla funcionalidad, para implementar nuestra aplicación, únicamente deberemos importar el paquete `javax.bluetooth.`

Como complemento a esta estructura vamos a añadir un diagrama con la arquitectura de esta librería junto con MIDP. Esta disposición puede verse en la figura 2.7. Conviene remarcar una característica importante de esta API. JSR82 no depende de MIDP y puede funcionar también sobre la edición estándar de java (J2SE).

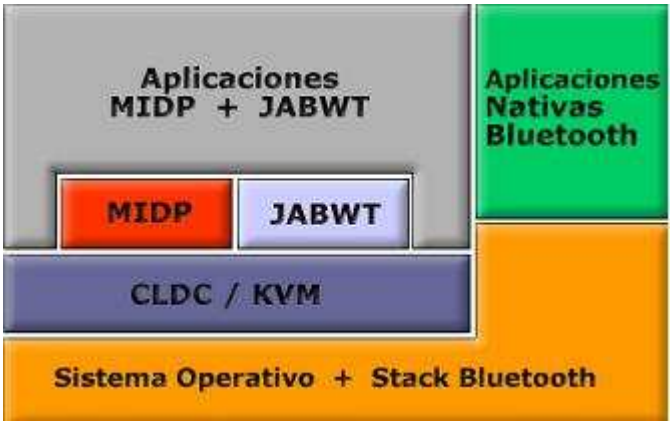

Figura 2.7: Arquitectura JSR82

Una característica que cabe destacar en este punto es el BCC (Bluetooth Control Center) que, si bien no es una clase o interfaz de esta librería, sí es una parte importante en lo que a seguridad se refiere. Habiendo múltiples aplicaciones ejecutándose concurrentemente, impide que unas puedan perjudicar a otras. El BCC está fijado por el proveedor y no puede ser modificado por ningún usuario.

Para terminar este apartado, mostraremos algunos diagramas sencillos sobre la puesta en funcionamiento de esta librería. Vamos a analizar su modo de ejecución para dar una idea sencilla y de conjunto. Las aplicaciones contemplan tres categorías de casos de uso.

La primera, que es la inicialización, es común para clientes, que descubren dispositivos y servicios, y servidores, que manejan clientes y ofrecen servicios[10]. La fase de inicialización es un proceso muy simple que viene detallado en la figura 2.9. Durante este proceso se obtiene una referencia al manejador de Bluetooth para, posteriormente, hacer que el dispositivo pueda ser detectado por otros. Por último creamos un DiscoveryAgent, el cuál proporcionará métodos de búsqueda de dispositivos y servicios.

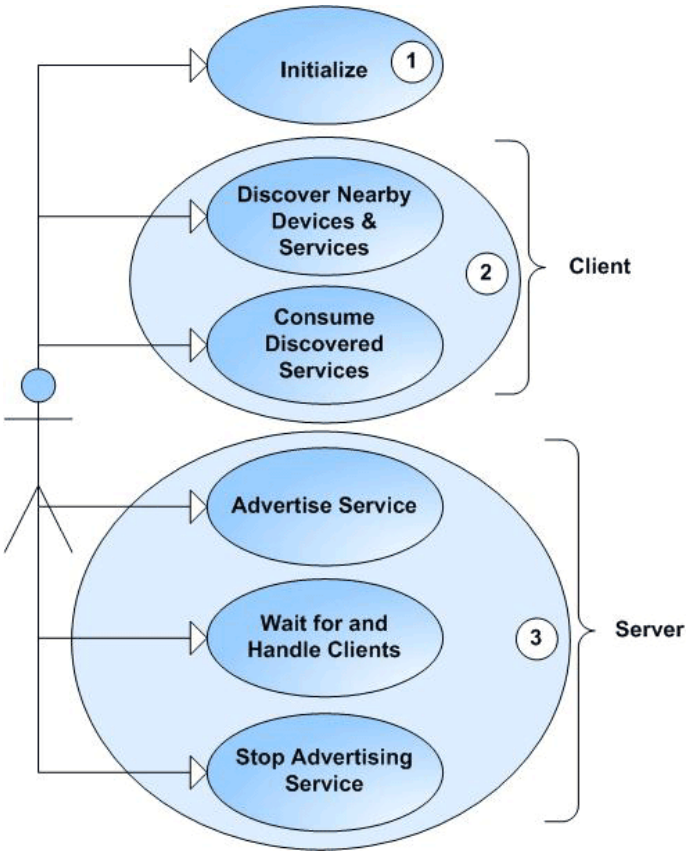

Figura 2.8: Casos de uso JSR82

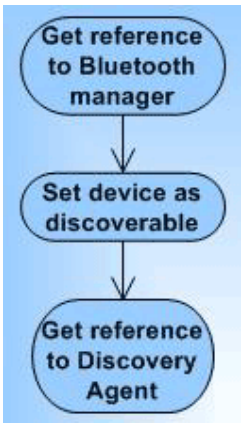

Figura 2.9: Fase de inicialización

Como último diagrama hemos considerado razonable incluir las actividades generales de cliente y servidor. La fase de inicialización, como hemos visto anteriormente, es común para ambos, desarrollando luego cada uno un tipo de funciones específicas, como pueden ser la creación de registros de servicio (Service Record) por parte del servidor o la búsqueda de dispositivos y servicios llevada a cabo por el cliente.

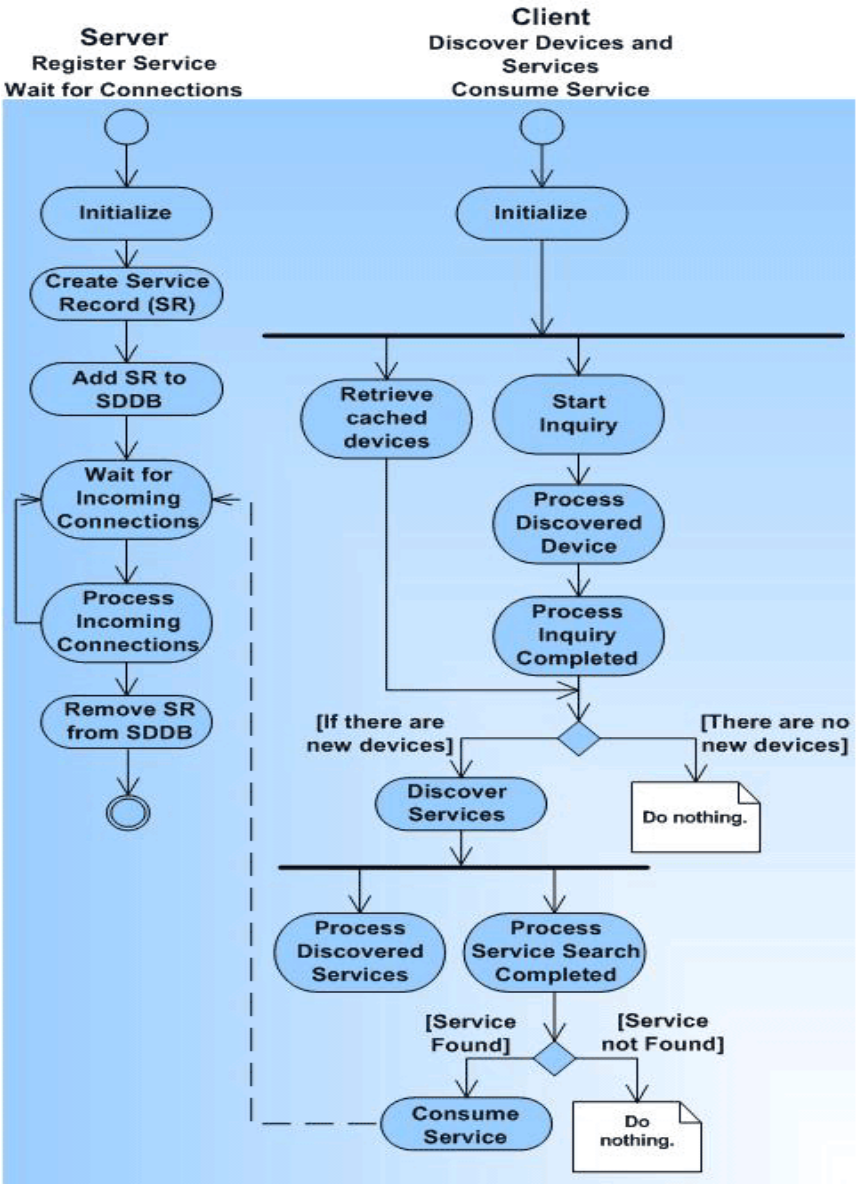

Figura 2.10: Funcionamiento general cliente y servidor

3 DESCRIPCIÓN DEL SISTEMA

En este capítulo veremos una descripción detallada y específica del funcionamiento de nuestro sistema de posicionamiento. Incluiremos información sobre el Midlet creado así como de las clases utilizadas y su funcionalidad para pasar, en el siguiente capítulo, a extender los datos aportados sobre la implementación de la aplicación Bluetooth, tanto cliente como servidor.

Como hemos visto en el capítulo introductorio el sistema creado aporta un funcionamiento sencillo. Se ha puesto especial interés en lograr esta sencillez y simplicidad para conseguir unos valores muy positivos en cuanto a consumo de baterías, coste y velocidad del sistema. Además se han utilizado unas convenciones de diseño que buscan constantemente el ahorro de la memoria, de por sí limitada, de los dispositivos.

En todo momento una de las principales premisas ha sido utilizar, única y exclusivamente, los recursos estrictamente necesarios para la consecución de nuestro proyecto. De este modo se ha conseguido un flujo mínimo de información entre dispositivos y un consumo de recursos de procesador y memoria limitados.

Este ahorro en el intercambio de la información ha sido posible gracias a un estudio detallado sobre el comportamiento de la librería utilizada. De esta forma se han utilizado recursos y clases fuera del ámbito habitual de las típicas aplicaciones cliente-servidor de JSR82 como veremos en el capítulo siguiente.

Teniendo en cuenta permanentemente estos puntos a tratar, considerados de vital importancia, el sistema resultante consiste, de manera genérica, en un cliente que peticiona información acerca de su localización y varios servidores o balizas que la reportan constantemente.

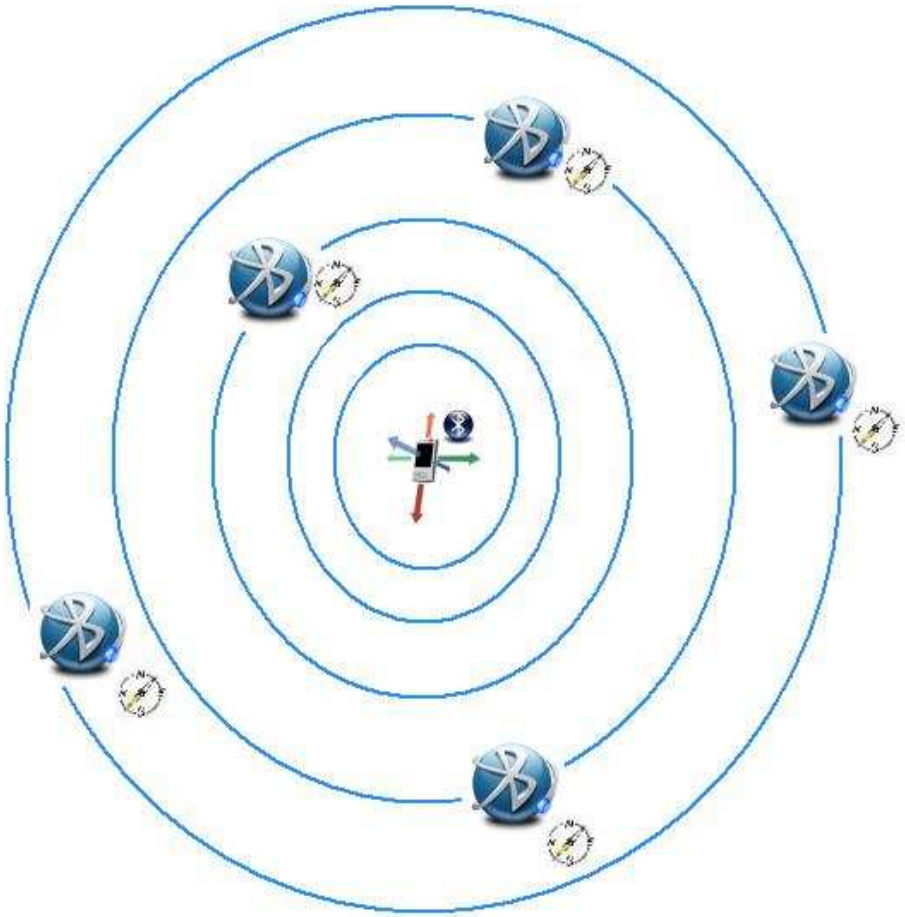

Figura 3.1: Esquema de sistema de localización

En este dibujo se orienta fácilmente al usuario sobre el modo de funcionamiento del sistema a alto nivel. Todas las balizas que estén dentro del radio de cobertura del cliente reportarán su correspondiente información de localización. Una vez se hayan obtenidos estos datos, el dispositivo tendrá información suficiente para calcular su localización mediante la ejecución de un determinado algoritmo matemático. Cabe remarcar que el dibujo anterior es una pequeña aproximación ilustrativa, si bien nuestro sistema puede actuar con varios clientes y servidores simultáneamente. Esto es posible gracias al diseño de las aplicaciones realizado y a las conexiones punto-multipunto que son soportadas por la tecnología Bluetooth.

Para lograr la mayor precisión posible se han diseñado varios tipos de algoritmos, creando, a su vez, una pequeña aplicación de simulación de los posibles errores medios de los mismos. El estudio realizado sobre estos cálculos matemáticos se mostrará ampliamente en el capítulo 5. Si bien, nuestro sistema implementa por defecto un mecanismo de cálculo de equidistancias entre las balizas que estén en la zona de cobertura para determinar la posición del dispositivo.

Con el fin de detallar ampliamente este modo de funcionamiento vamos a describir un hipotético caso de uso del sistema en el que nos encontráramos con un par de servidores.

Tomando coordenadas cartesianas y como unidad de medida el metro, las ubicaciones de nuestros servidores podrían ser (1,4) para la baliza 1 y (7,6) para la baliza 2. Con esto tenemos que el sistema daría una posición de localización para nuestro dispositivo de (4,5) como podemos apreciar en la figura 3.2.

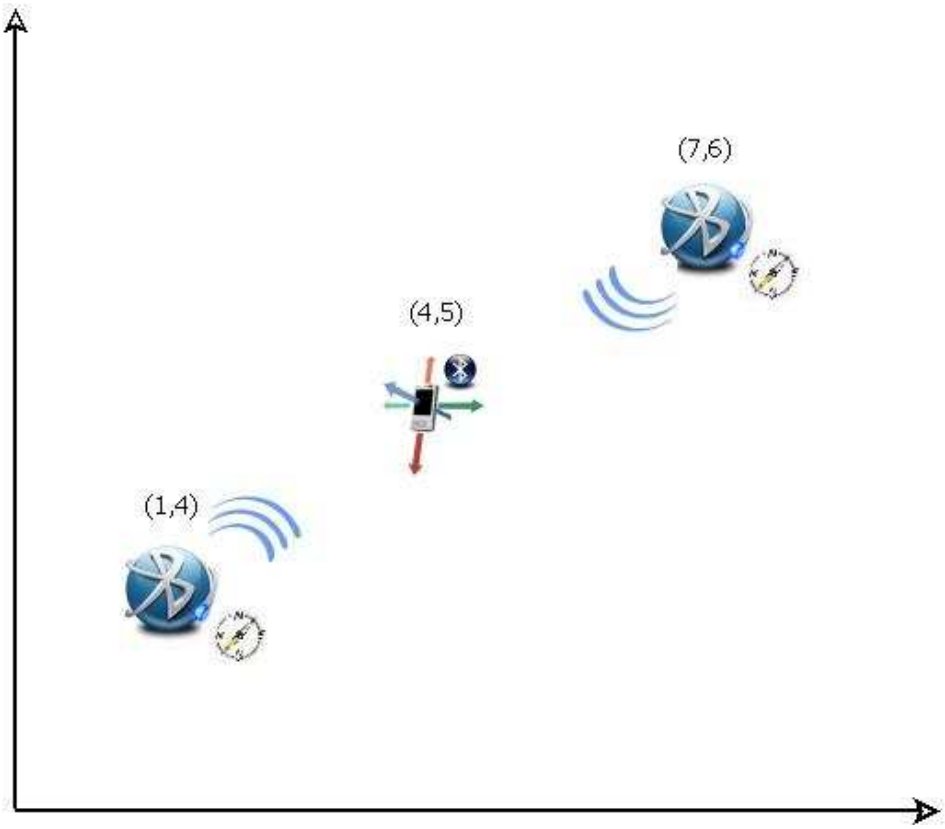

Figura 3.2: Ejemplo de sistema de localización

Como es lógico imaginar, con estos modos de funcionamiento, la precisión aumenta proporcionalmente con el número de balizas en la zona de cobertura. Retomando el ejemplo del dibujo anterior tendríamos escasa información de posicionamiento, con la consiguiente pérdida de precisión dependiendo del punto donde nos ubicáramos. Por otra parte, debido al propio algoritmo, los servidores más alejados atraerían más hacia sí la posición final que los puntos más cercanos al eje de referencia.

Mediante el análisis del sistema a bajo nivel, comprobamos cómo interaccionan cliente y servidor para lograr ese intercambio de información de localización descrito anteriormente en este capítulo. Se ha creado un servicio específico para este fin con un atributo de servicio que proporciona al cliente dicha información. Este es un comportamiento genérico en cuanto a conexiones Bluetooth se refiere.

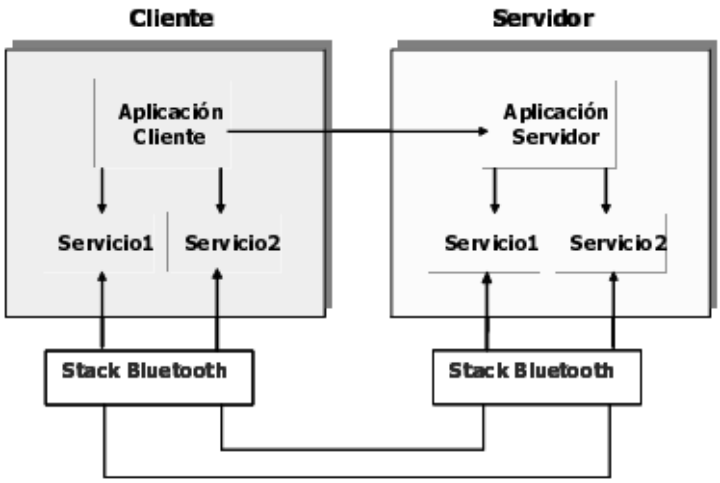

Figura 3.3: Conexión cliente-servidor

Para completar la descripción gráfica de lo mencionado anteriormente incluiremos unos diagramas de secuencia que muestran la interacción entre el usuario, la aplicación cliente y la aplicación servidora. Vamos a contemplar dos posibles casos:

1. **CASO1**: Dispositivo servidor ofrece información de localización.

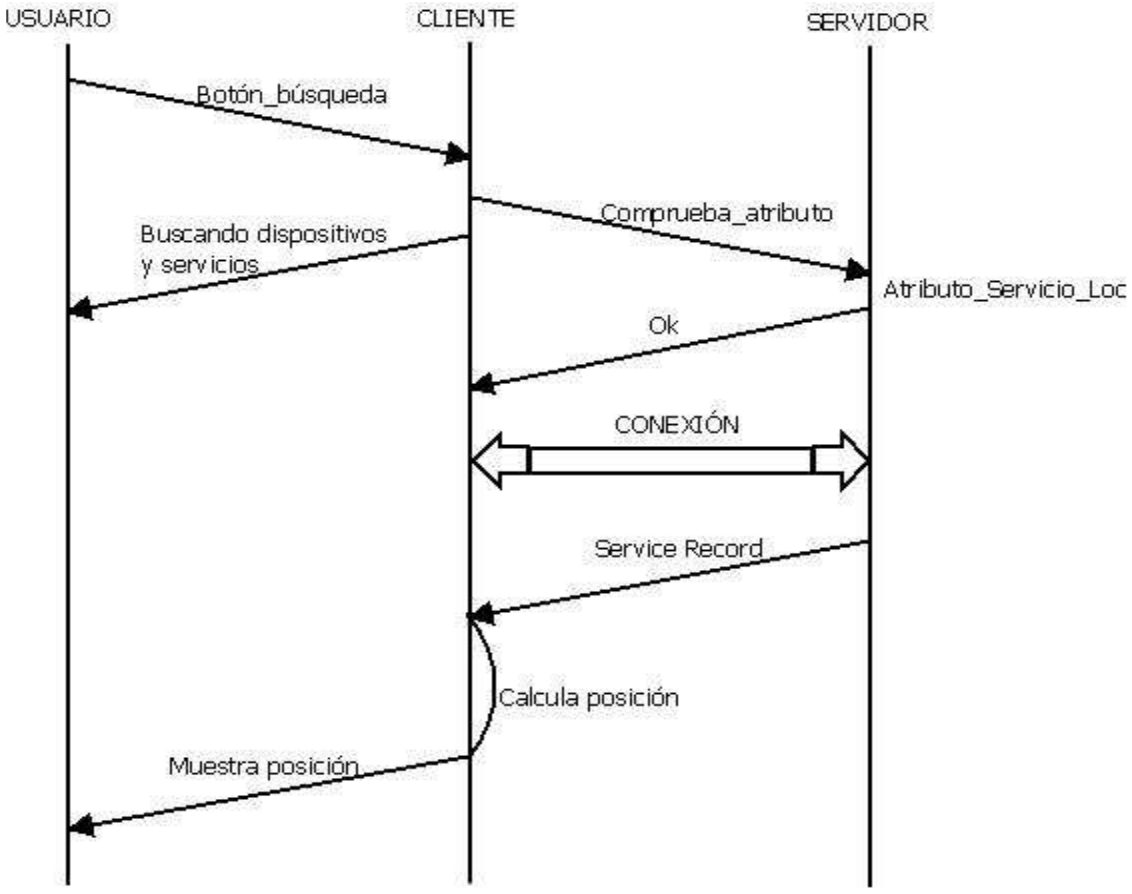

Figura 3.4: Diagrama de secuencia con información de localización

2. **CASO2:** Dispositivo servidor no puede ofrecer información de localización.

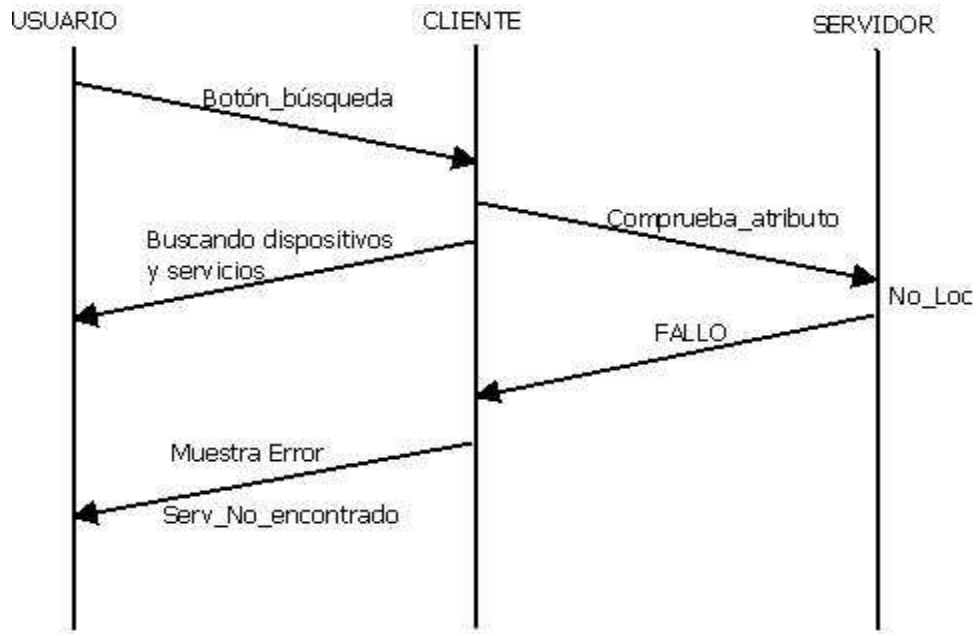

Figura 3.5: Diagrama de secuencia sin información de localización

El punto de inicio de nuestro sistema, como introdujimos en el capítulo 1, es un Midlet que nos da a elegir entre dos modos de ejecución, cliente y servidor. Esta clase inicial únicamente sirve como arranque de la aplicación, muestra una breve introducción y automáticamente pasa a la pantalla de selección de modo de funcionamiento de nuestro sistema. Esta elección entre cliente o servidor hará que se lance una u otra aplicación Bluetooth en nuestro dispositivo.

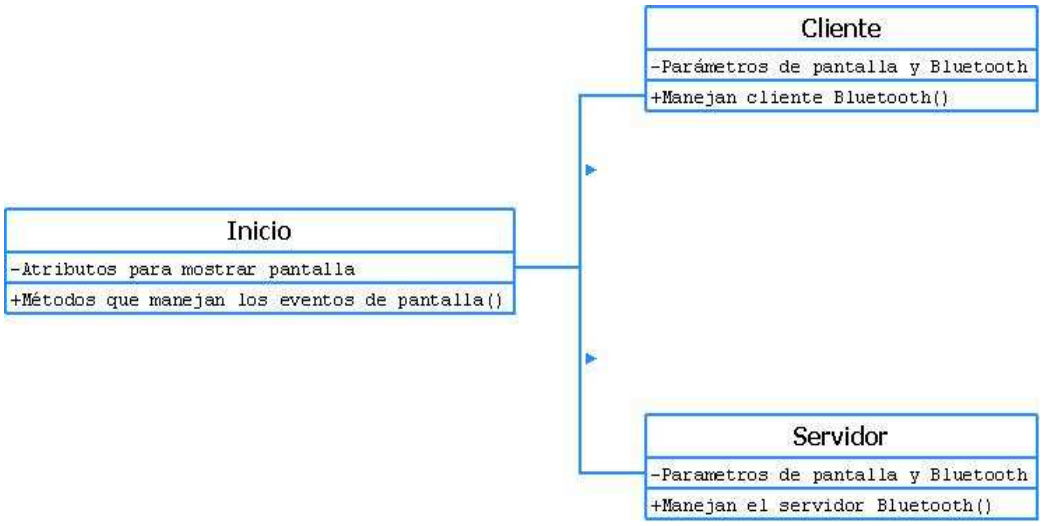

Figura3. 6: Diagrama de clases del sistema

La sencillez de ejecución viene reflejada en el diagrama de clases de la figura 3.3. Tanto métodos como atributos se han establecido de manera genérica para pasar a una mayor descripción en el siguiente capítulo.

Por último, un apunte en cuanto al entorno de ejecución de nuestro sistema se refiere. Destacaremos que su funcionamiento es válido tanto en exteriores como en interiores. En el segundo capítulo de la memoria vimos las coberturas máximas a las que la tecnología Bluetooth nos permitía llegar sin ningún obstáculo presente (<20m) o encontrándonos alguno (<10m). Con estas pruebas realizadas podemos plantearnos varios tipos de entornos de utilización del sistema. Además, como hemos podido observar en algunas de las pruebas realizadas, puede haber conexión entre dos dispositivos en habitaciones contiguas, siempre y cuando no se supere una distancia máxima de entre 7 y 9 metros. Realizadas estas comprobaciones podremos plantear la ejecución de nuestro sistema en sitios tan dispares como pudieran ser un museo, un parque de atracciones, un centro comercial, el casco histórico de una ciudad o, porqué no, una mansión, siempre y cuando la disposición de las balizas sea adecuada.

4 DESCRIPCIÓN DE LA IMPLEMENTACIÓN

Llegados a este punto, vamos a detallar profunda y estructuradamente la aplicación al completo. Veremos el diseño y funcionamiento desde el Midlet inicial hasta el programa creado para el testeo de los errores. A lo largo de este capítulo, el análisis de toda la implementación se acompañará con descripciones gráficas en forma de diagramas de secuencia, de flujo, esquemas y demás complementos que se estimen necesarios. Para comprender mejor este diseño se comenzará dando un breve repaso a las clases más importantes de la librería utilizada, para luego comenzar con la descripción de la implementación.

4.1 CLASES PRINCIPALES DE JSR82

Una breve explicación genérica de las principales clases utilizadas de esta librería será proporcionada a lo largo de este apartado. Para cada una se incluirán algunos de los métodos más utilizados. Todas ellas se encuentran en el paquete `javax.Bluetooth`. Gracias a esta descripción podremos entender mejor el porqué de la implementación realizada.

`DiscoveryAgent`

Podríamos considerar esta clase como la principal de nuestra aplicación. Es muy importante en el desarrollo de todo el sistema ya que se encarga de las búsquedas de dispositivos y servicios por parte del cliente así como de su cancelación. Obtenemos una instancia a esta clase mediante un método de `LocalDevice`, que veremos más adelante.

```
agent = localDevice.getDiscoveryAgent();
```

Los métodos más importantes son, `startInquiry()`, que permite la búsqueda de dispositivos Bluetooth, `searchServices()` que realiza lo propio con los servicios, así como `cancelInquiry()` y `cancelServiceSearch()` que cancelan estas búsquedas.

`DiscoveryListener`

Al igual que la anterior clase, este interfaz es de vital importancia para nuestra aplicación. Con ella tenemos escuchadores para cualquier evento que pueda ocurrir en las búsquedas de dispositivos o servicios realizadas por `DiscoveryAgent`. Además, nos reporta información sobre el estado de las búsquedas y sus resultados.

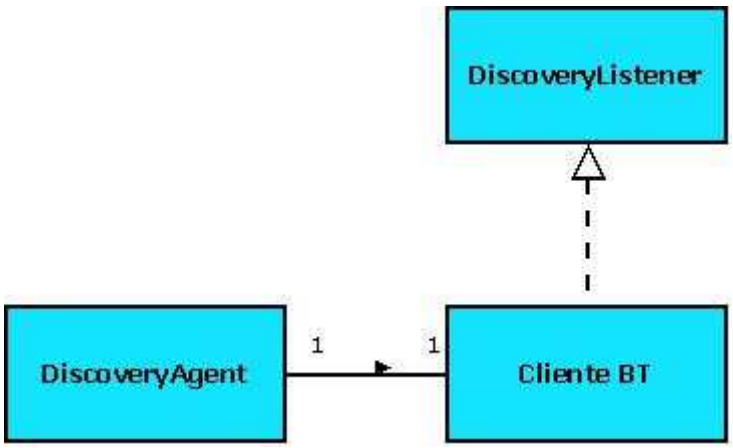

Figura 4.1: Relación entre DiscoveryAgent y DiscoveryListener

Este interfaz nos proporciona cuatro métodos a implementar, dos para las búsquedas de dispositivos y dos para las de servicios. Tienen un funcionamiento similar, siendo `deviceDiscovered()` y `servicesDiscovered()` llamados cuando se ha encontrado un solo dispositivo o servicio respectivamente e `inquiryCompleted()` y `serviceSearchCompleted()` llamados cuando la búsqueda de todo dispositivo o servicio se ha completado.

`LocalDevice`

Mediante la utilización de esta clase podemos obtener funciones para controlar y acceder a nuestro dispositivo. Seremos capaces de conseguir, entre otras funciones, la dirección Bluetooth, poner nuestro dispositivo local en modo alcanzable para otros dispositivos, obtener el `ServiceRecord` de una conexión dada o conseguir el `DiscoveryAgent` con los métodos `getBluetoothAddress()`, `setDiscoverable()`, `getRecord()` y `getDiscoveryAgent()` respectivamente.

`RemoteDevice`

La principal función de esta clase en nuestro sistema es la de representar al servidor en la aplicación cliente. Mediante el método `getBluetoothAddress()` podemos obtener la dirección del servidor.

`ServiceRecord`

Esta interfaz representa un registro o contenedor de las características de un determinado servicio. Está estructurado mediante pares de atributos junto con sus valores (ID_atributo, valor del atributo). Esta clase es muy importante ya que mediante el método `getAttributeValue()` podemos obtener el `DataElement` con la información de localización.

`DataElement`

Toda vez que pese a la sencillez de esta clase la mayoría de las mejoras, con respecto a otras aplicaciones Bluetooth, que nuestro sistema incorpora son gracias a ella, vamos a verla con más detalle en un apartado posterior.

Esta clase proporciona los tipos de datos y valores que puede tomar cualquier atributo de servicio. Nos interesaremos, sobre todo, en el tipo `String`. Con el método `getValue()` podremos obtener el valor del atributo.

4.2 MIDLET INICIO

En anteriores capítulos ya se adelantó el funcionamiento de este sencillo Midlet que actúa como “lanzador” de los dos diferentes modos de funcionamiento de la aplicación, cliente y servidor.

Una vez ha mostrado la pantalla inicial de introducción, aparece el menú de selección para que la aplicación corra en un modo u otro y espera a que el usuario introduzca su elección, cliente o servidor.

Esta clase, independientemente de la escucha de eventos, también aportará funcionalidad relacionada con la aparición en pantalla de varios mensajes de error o información así como la actualización de la misma.

4.3 CLIENTE

Una vez el usuario ha elegido esta opción, la aplicación comenzará a actuar como un cliente Bluetooth. Esta clase es el núcleo de nuestro sistema ya que es la que va a aportar la funcionalidad por la que fue creado.

Mediante este modo de uso el usuario podrá disponer cuando lo estime conveniente de su localización en un determinado espacio, siempre y cuando lo permita la cobertura proporcionada por la tecnología utilizada.

Comenzaremos la descripción de nuestro cliente con un resumen aproximativo del comportamiento del mismo. La figura 4.2 ilustra gráficamente, abstrayéndonos del código, cómo se comporta el cliente y la funcionalidad que aporta.

El diagrama planteado sigue un orden de secuencia sencillo e intuitivo. Una vez que el usuario requiere información sobre su posicionamiento, el cliente obtiene los datos de todos los servidores dentro de su rango de cobertura. Hecho esto, cruza los datos utilizando un algoritmo predefinido para finalmente mostrar la posición obtenida.

A lo largo de esta memoria se ha venido diciendo que nuestro sistema utiliza para comunicarse el protocolo RFCOMM, el cuál está estrechamente ligado al perfil de puerto serie (SPP). Debido a esto, los pasos a seguir por cualquier cliente para establecer una conexión estándar RFCOMM con un servidor son los siguientes:

- Buscar servicios para obtener un `ServiceRecord`
- Mediante el SR obtenido, construir una conexión
- Abrir una conexión con el servidor
- Recibir y enviar datos

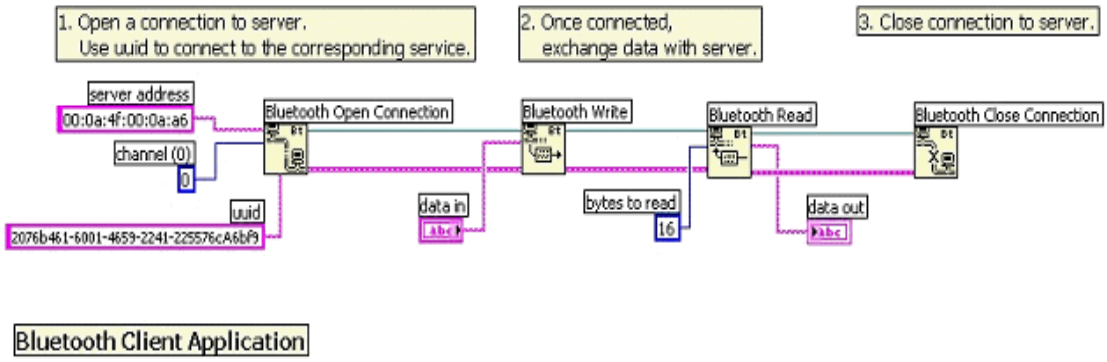

Figura 4.2: Conexión cliente estándar

Estos serían los puntos de acción necesarios en una conexión típica cliente-servidor utilizando este protocolo, si bien, como ya se ha comentado anteriormente, nuestro sistema incorpora novedades en las pautas de implementación que permiten dotarlo de una funcionalidad más eficiente. En concreto, en nuestro cliente solamente es necesario el primer paso, buscar el servicio de localización en los servidores para poder obtener el registro. La información de localización que el cliente requiere se encuentra en el `ServiceRecord` obtenido y por ello no es necesario crear una conexión para la recepción de datos. Debido a la heterodoxia de nuestro sistema, esta novedad será tratada en un apartado posterior.

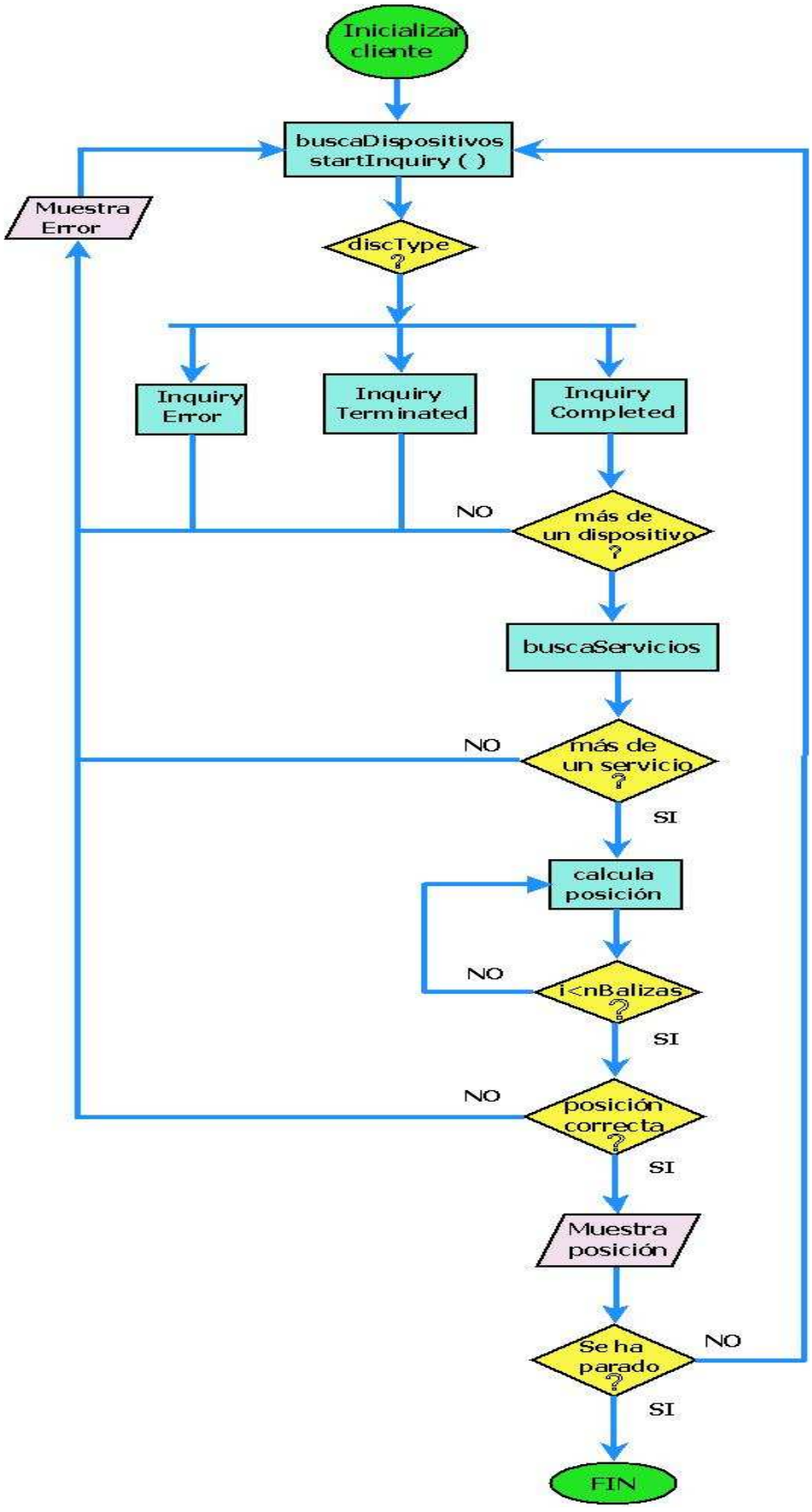

Figura4. 3: Diagrama de flujo Cliente

Una vez que se han creado e inicializado los botones y parámetros que manejan la interfaz gráfica del cliente, se lanza un hilo que no se parará hasta que el usuario salga de la aplicación.

Primeramente, dicho hilo, obtendrá el `DiscoveryAgent` necesario en todo caso para las búsquedas futuras. Una vez obtenido, se inicializarán todos los parámetros referentes al servicio y protocolo utilizado.

Llegados a este punto, la aplicación cliente está en condiciones de poder realizar las búsquedas requeridas por el usuario, por lo que permanece en un estado de espera hasta que el usuario necesite su información de localización. Cuando esto ocurre, realiza inicialmente una búsqueda de todos los dispositivos Bluetooth que pueda tener el terminal dentro de su rango de cobertura mediante `startInquiry()`.

```
agent.startInquiry (DiscoveryAgent.GIAC, this);
```

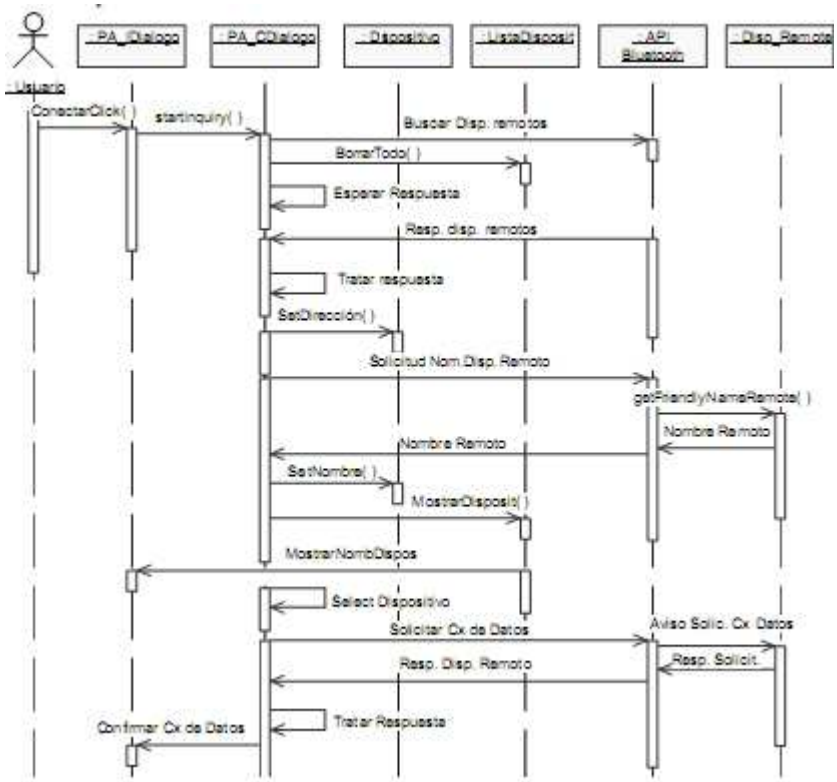

Figura4. 4: Descubrir dispositivos

En la figura 4.4 se muestra el intercambio de mensajes entre las entidades que intervienen en el proceso de descubrimiento de dispositivos y JSR82.

Si la aplicación no ha sido parada por el usuario a través del correspondiente botón creado en pantalla para tal fin, realiza las comprobaciones pertinentes sobre los resultados de la búsqueda una vez terminada (completa, con errores…).

Es necesario que se hayan detectado al menos dos dispositivos para poder cruzar la información de localización y aportar un resultado fiable. Si no es así se mostrará el error por pantalla y comenzará de nuevo la búsqueda cuando el usuario lo estime conveniente. Cuantos más servidores sean capaces de ofrecer datos de posicionamiento más precisión obtendremos en los resultados.

Encontrados correctamente los dispositivos, el siguiente paso será buscar el servicio de posicionamiento, creado con anterioridad, en los servidores hallados mediante `searchServices()`, método de `DiscoveryAgent`.

```
agent.searchServices (attrSet, uuidSet, r, this);
```

Por cada servidor que pueda ofrecer información de localización obtiene su correspondiente `ServiceRecord` y de éste, el `DataElement` que contiene los datos de posicionamiento necesarios para nuestro sistema. Una vez obtenido y almacenado pasa a comprobar si el servicio necesario está incluido en el siguiente dispositivo encontrado.

```
public void servicesDiscovered (int transID, ServiceRecord[]
       servRecord) {

  registro.addElement (servRecord[0]);

  DataElement d = servRecord[0].getAttributeValue(ID_ATRIBUTO_LOC);

  posiciones.addElement(d.getValue());

}
```

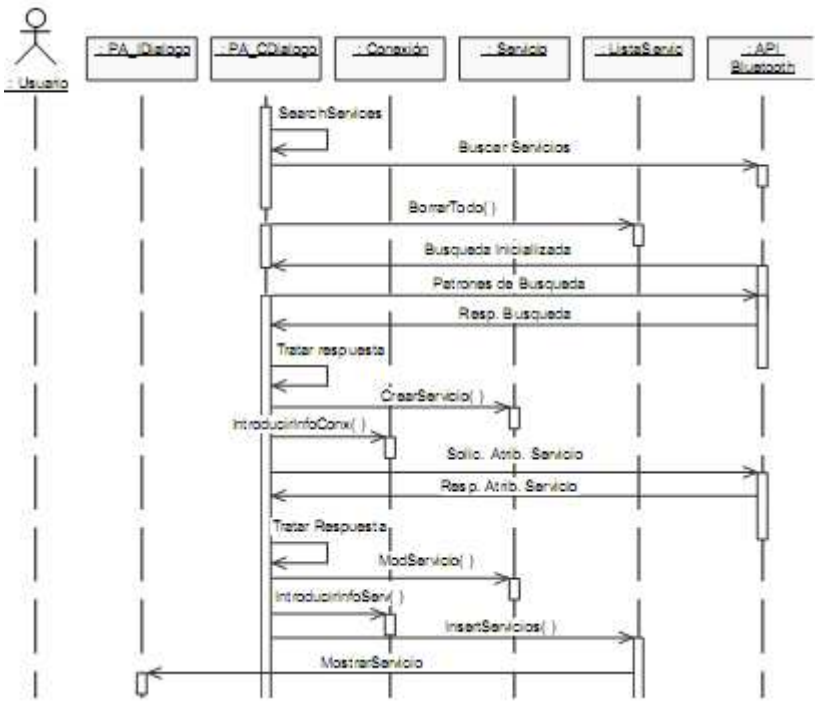

Figura4. 5: Descubrir servicios

Los mensajes necesarios para la búsqueda de servicios, así como para obtener el atributo correspondiente de servicio, que reportará la información de localización necesaria, aparecen en el diagrama de secuencia de la figura 4.5.

Como en el caso de los dispositivos, es necesario encontrar al menos dos servicios para poder ofrecer al usuario unos resultados mínimamente aceptables.

Una vez se ha completado la búsqueda de servicios, para cada servidor o baliza, se obtiene su posición almacenada y se va sumando a cada coordenada, para finalmente dividir entre el número de balizas. De esta forma se calcula un punto equidistante como posición del usuario entre todos los dispositivos encontrados. Como se ha comentado en capítulos anteriores, el algoritmo implementado puede variar, y en base a esto, los errores medios entre la posición obtenida y la real son diferentes.

A la vez que se realizan los cálculos de posicionamiento, para cada baliza se comprueba la validez del formato introducido. En caso de que este formato, en cualquiera de los servidores, no sea el adecuado o no se hayan introducido números, la aplicación volverá al paso inicial de espera de búsqueda de dispositivos sin mostrar información de localización alguna. Únicamente mostrará un mensaje informativo sobre el error de formato ocurrido.

Si por el contrario no ha habido ningún error en cuanto al formato, la posición obtenida se considerará correcta y será mostrada por pantalla al usuario. Mostrada la localización que se ha calculado, el programa volverá a su estado inicial de espera de eventos, es decir, el usuario deberá pulsar de nuevo el botón de búsqueda para obtener una nueva localización y comenzará de nuevo a ejecutarse de la manera explicada.

Si en cualquier momento de la ejecución el usuario decidiera terminar, el hilo se eliminaría y la aplicación pasaría a mostrar la pantalla de inicio en la que el usuario puede elegir funcionalidad cliente o servidor.

4.4 SERVIDOR

Este es el modo de ejecución que será utilizado para todas las balizas que puedan aportar información de localización a los usuarios. Cada una de estas balizas tendrá almacenada una posición específica que enviará al cliente cuanto éste se conecte.

Como en el caso del cliente, consideramos apropiado comenzar la descripción de la aplicación servidora con un resumen de su funcionamiento a grandes rasgos. Los principales pasos a seguir son ejecutados por todas las aplicaciones servidor Bluetooth estándar.

Una vez se han inicializado los parámetros necesarios, el servidor espera a que se le introduzcan los correspondientes datos de posicionamiento para almacenarlos. Hecho esto, ya está en disposición de ofrecérselos a cualquier cliente que lo requiera, dentro del rango de cobertura.

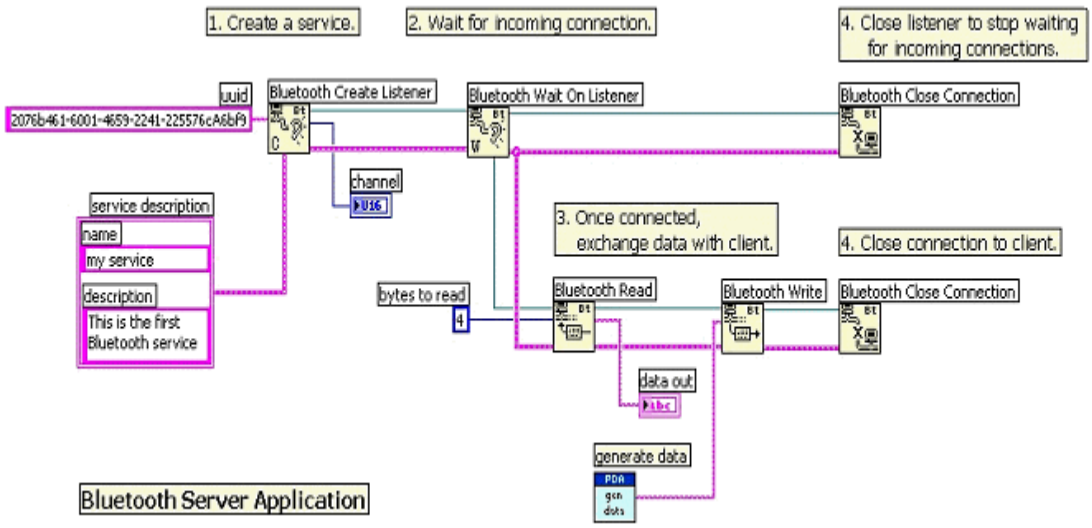

Figura4. 6: Conexión servidor estándar

Una aplicación servidora se caracteriza por su capacidad de ofrecer determinados servicios a sus clientes. En el API utilizado estos servicios están definidos por el correspondiente registro de servicio, luego como pasos iniciales para cualquier servidor necesitaremos trabajar con los denominados `ServiceRecord`.

- Crear SR para el servicio que necesitemos dar disponibilidad
- Añadir el SR a la SDDB
- Registrar el servicio

A su vez, debido a la utilización de RFCOMM para cualquier tipo de conexión, el servidor deberá seguir unos pasos característicos:

- Construir URL para indicar cómo conectarse al servicio
- Registrar la URL en el SR
- Hacer disponible el SR al cliente
- Aceptar conexiones de clientes
- Enviar y recibir datos del cliente

La URL por la cual cualquier cliente podrá conectarse al servicio de localización ofrecido tiene el formato siguiente:

```
btspp://localhost: FEDCBA00000000000000000999999999
;name=Servidor_Localizacion;authorize=false
```

Mediante el parámetro authorize queremos denotar que cualquier cliente tendrá acceso a la aplicación, para ser consecuentes con nuestra política de seguridad. Este parámetro tiene un valor “false” por defecto. Para hipotéticos usos seguros se podrá restringir este acceso, encriptar las comunicaciones o autenticar al cliente.

El funcionamiento completo de nuestra aplicación servidora viene detallado en el diagrama de flujo de la figura 4.7.

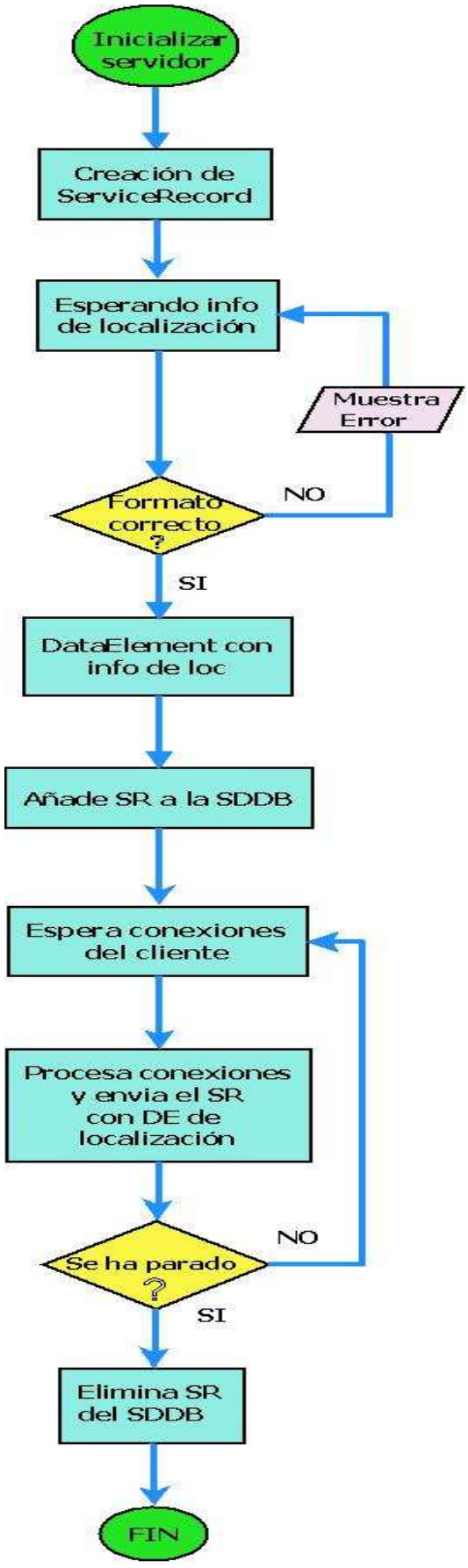

Figura4. 7: Diagrama de flujo servidor

El primer paso necesario, al igual que en el cliente, es la creación de los parámetros de pantalla correspondientes, así como los escuchadores de los botones de la misma que nos permitan interaccionar con la aplicación.

Como en el cliente, nuestro servidor será manejado por un hilo mediante el cual aceptaremos las conexiones que pudieran entrar hasta que el usuario decida parar el servidor.

Dicho hilo establece el dispositivo como detectable, incorpora la URL necesaria para el servicio y crea el `ServiceRecord` correspondiente. Una vez se han realizado estos pasos previos, el servidor está listo para recibir la información de posicionamiento que quedará almacenada.

Cuando el usuario ha introducido los datos correspondientes, la aplicación comprobará, primeramente, si el formato introducido es el correcto. Estos son los formatos aceptados:

X=x0, Y=y0, Z=z0
X=x0 Y=y0 Z=z0

Es aceptado el formato tanto en mayúsculas como en minúsculas. Un formato en el que falte alguna coordenada, aunque las restantes sean correctas, se considerará erróneo así como aquél en el que las coordenadas no se encuentren separadas por espacios o comas. Como es lógico, si en vez de coordenadas numéricas se introduce cualquier otro tipo de parámetro el formato se tomará como incorrecto.

En caso de que el formato no sea aceptado, aparecerá un mensaje de formato erróneo y se mostrará de nuevo la pantalla de inserción de datos. Si por el contrario el formato es el esperado se tomará como la posición real del servidor. Estos datos son lo necesarios para que nuestro sistema funcione correctamente. Una vez obtenidos, son almacenados en un `DataElement` (d) de tipo `String`. Éste será incluido en el `ServiceRecord` creado anteriormente.

```
registro.setAttributeValue(ID_ATRIBUTO_LOC, d);
```

Si este proceso se ha llevado a cabo sin problemas, el servicio ya se encuentra activo y con su información de localización insertada correctamente. El siguiente paso consiste en añadir el registro de servicio a la base de datos de descubrimiento de servicios (SDDB). Hecho esto, nuestro servidor estará en disposición de ofrecer su información a cualquier cliente que establezca una conexión Bluetooth con él.

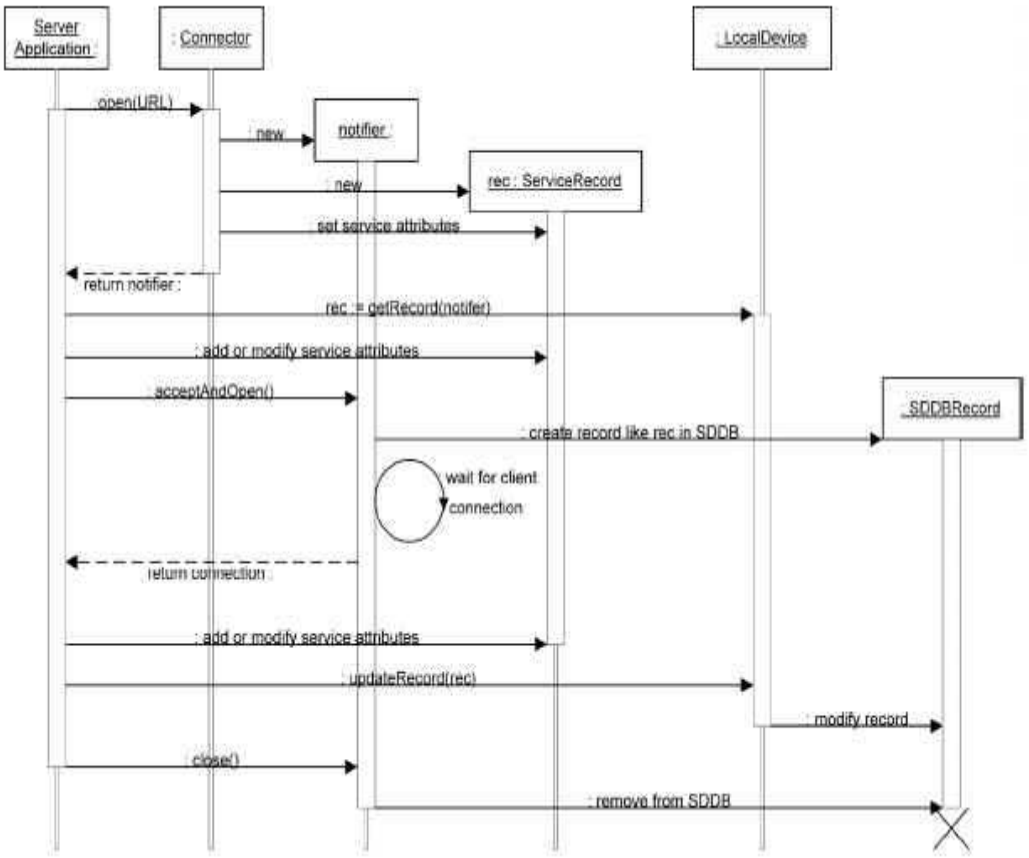

Figura4. 8: Registro de servicios

Como se puede apreciar, mediante la llamada a `Connector.open()`con la URL determinada, se crea el `ServiceRecord` correspondiente al servicio establecido. Éste es añadido a la SDDB al realizar la llamada al método `acceptAndOpen()`. Dicho registro se obtiene mediante el método `getRecord()`.

Nuestra aplicación cliente funcionará de esta forma con todas las conexiones entrantes hasta que el usuario decida pararla, momento en el cual el registro de servicio ya no es necesario por lo que se elimina de la SDDB mandando un `close()` al `notifier` y eliminando el hilo creado anteriormente. Una vez se ha cerrado correctamente la aplicación volverá a mostrar la pantalla inicial de selección.

4.5 NOVEDAD DE LA IMPLEMENTACIÓN (DataElements)

En los apartados anteriores se ha descrito el funcionamiento tanto de la aplicación cliente como servidora. A su vez se han mostrado las ligeras diferencias que nuestro sistema puede aportar en relación a otros que siguen unas pautas claramente definidas.

Para el caso de una aplicación cliente habitual, como se ha podido ver anteriormente, los pasos a seguir son:

- Buscar servicios para obtener un `ServiceRecord`
- Mediante el SR obtenido, construir una conexión
- Abrir una conexión con el servidor
- Recibir y enviar datos

De esta forma un ejemplo de código típico para un cliente siguiendo estos pasos podría ser el siguiente:

```
...
// (Ya se ha obtenido SR)
String url =record.getConnectionURL(record.NOAUTHENTICATE_NOENCRYPT, false);

// Abre connexion al servidor
StreamConnection connection = (StreamConnection) Connector.open(url);

// Enviar/Recibir datos
try {
    byte buffer[] = new byte[100];
    String msg = "hello there, server";
    InputStream is = connection.openInputStream();
    OutputStream os = connection.openOutputStream();

    // Enviar datos al servidor
    os.write(msg.getBytes);

    // Leer datos del servidor
    is.read(buffer);
    connection.close();

} catch(IOException e) {
  e.printStackTrace();
}
...
```

Si tenemos ahora en cuenta una aplicación servidora, su forma de funcionamiento habitual consta de los siguientes pasos característicos:

Construir URL para indicar cómo conectarse al servicio
Registrar la URL en el SR
Hacer disponible el SR al cliente
Aceptar conexiones de clientes
Enviar y recibir datos del cliente

Debido a estos pasos esenciales, cualquier servidor deberá presentar en algún momento un fragmento de código como el que sigue a continuación:

```
...
// (Ya se ha obtenido UUID)
String ServiceURL = "btspp://localhost:10203040607040A1B1C1DE100;
name=SPP Server1";
try {
    // Crea conexión
    StreamConnectionNotifier notifier =
       (StreamConnectionNotifier) Connector.open(serviceURL);

    // Aceptar conexiones de clientes
    StreamConnection connection = notifier.acceptAndOpen();

    byte buffer[] = new byte[100];
    String msg = "hello there, client";
    InputStream is = connection.openInputStream();
    OutputStream os = connection.openOutputStream();

    // Enviar datos al cliente
    os.write(msg.getBytes());

    // Leer datos del cliente
    is.read(buffer);
    connection.close();

} catch(IOException e) {
  e.printStackTrace();
}
...
```

En el caso de nuestro servidor, los primeros pasos de creación de conexión y manejo de peticiones de clientes son semejantes, si bien, posteriormente no es necesario crear Streams de datos para intercambiar información. De esta forma nuestro cliente quedaría del siguiente modo, más sencillo y funcional:

```
...
// (La connexion ya se ha creado)
while (seguir) {

     try {

          if(seguir)
            con = notifier.acceptAndOpen ();

          }

     catch (IOException e) {

           // Si hay error con este cliente pasamos a la siguiente
          // conexión
         continue;
        }

}
...
```

Esto es posible gracias a un proceso de investigación y análisis previo, mediante el cual se consiguió dar a los `DataElements` un rol en nuestro sistema diferente a las típicas implementaciones creadas con anterioridad. En nuestra aplicación servidora, el paso siguiente a la introducción de los datos de posicionamiento por parte del usuario, no es otro que el almacenamiento en los mencionados `DataElements` de esta información. Por tanto únicamente ha sido necesario diseñar el siguiente método:

```
public void establecerLocalizacion(){

      //Atributo de servicio con la información de señalización
      DataElement d = new DataElement(DataElement.String,(Object)localizacion)

      registro.setAttributeValue (ID_ATRIBUTO_LOC, d);

      synchronized(this){

         notify();

     }
}
```

En el caso del cliente, la mejora es más sustancial. De este modo, una vez obtenido el `ServiceRecord` (paso 1), el cliente ya posee toda la información que necesita para realizar sus cálculos de posicionamiento y no son de vital importancia los pasos posteriores. Es por esto que no es necesario construir una conexión con el servidor para la recepción de datos. Toda la información necesaria se encuentra contenida en el `DataElement` correspondiente. Con este sencillo modo de funcionamiento, únicamente será necesario implementar de una forma especial el método de `DiscoveryListener`, `servicesDiscovered()`.

```
    public void servicesDiscovered (int transID, ServiceRecord[] servRecord) {

            registro.addElement (servRecord[0]);

            DataElement d = servRecord[0].getAttributeValue(ID_ATRIBUTO_LOC);

            posiciones.addElement(d.getValue());

    }
```

En la figura 4.9, en la que aparece reflejado el sistema completo, se puede apreciar la interacción entre cliente y servidor. Es en la búsqueda de servicios, cuando se ha encontrado el servicio correspondiente al posicionamiento, donde se obtiene el DataElement que proporcionará la información de localización.

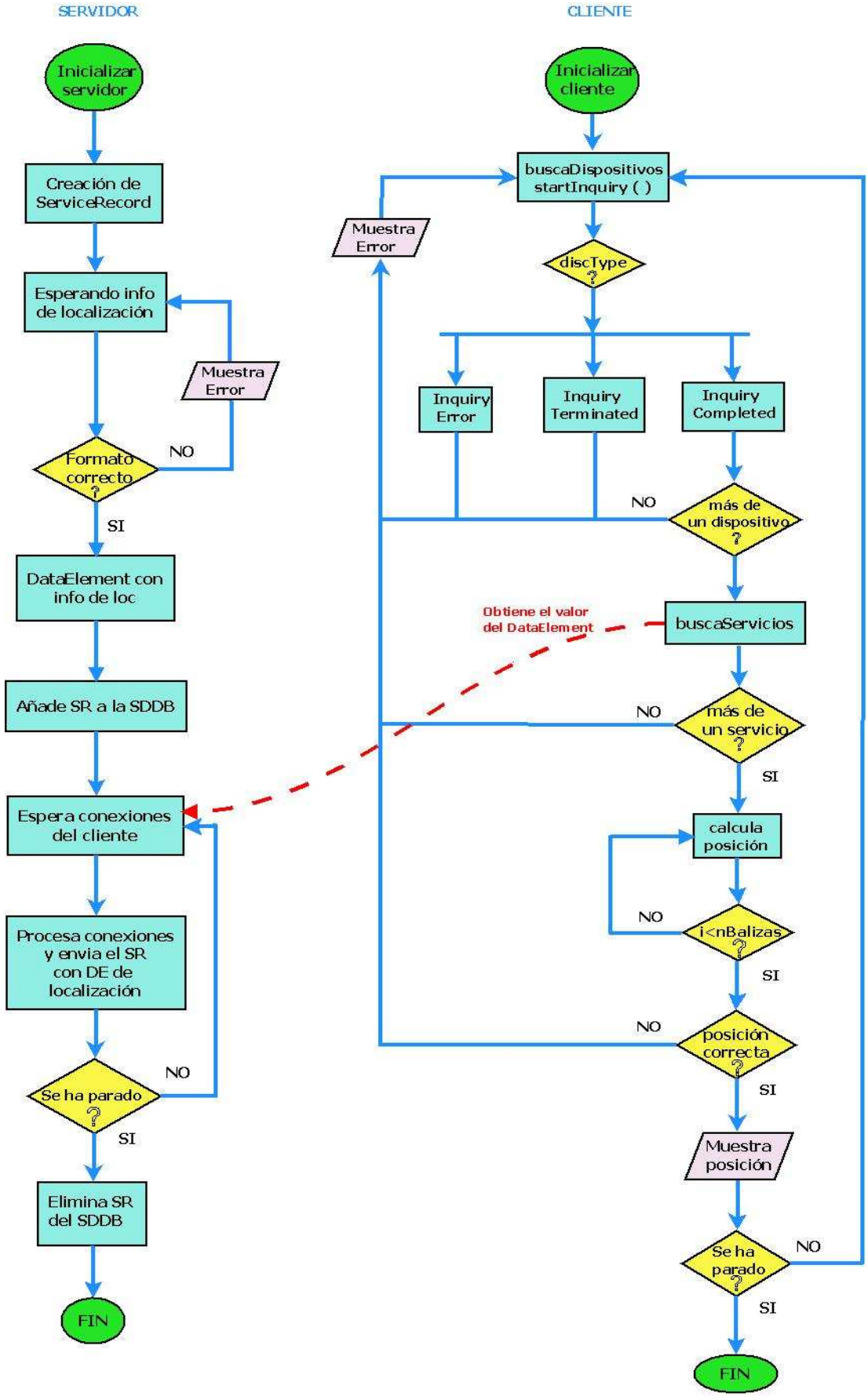

Figura 4.9: Diagrama de flujo sistema completo

Debido a estas novedades, estamos en disposición de afirmar que el sistema creado aportará una serie de importantes mejoras. Nuestro sistema será capaz de consumir menos memoria, debido a su ahorro de código. Este ha sido un punto de acción prioritario en la consecución de nuestro proyecto dado que es un recurso muy limitado en los dispositivos móviles. A su vez, el procesador tendrá menos sentencias que ejecutar por lo que se reducirá su carga de trabajo. Por otra parte, el intercambio de información entre dispositivos, mediante Bluetooth, se reduce a la obtención del correspondiente registro de servicio por parte del cliente.

Del mismo modo, dado el carácter móvil del usuario, un sistema de este tipo será mejor cuanto menos tarde en ofrecer una localización, ya que al cliente se le presupone movimiento. Al no tener que implementar Streams para el intercambio de información, el tiempo de ejecución de nuestro sistema disminuye drásticamente. Por otra parte, en caso de que el terminal pierda la conexión Bluetooth, nuestro sistema no se verá afectado una vez posea el DataElement correspondiente. Esta es otra aportación importante a resaltar.

De esta forma, conseguimos un sistema rápido, eficiente, portable, robusto y ligero, válido para cualquier dispositivo móvil capaz de soportar J2ME.

5 SIMULACIÓN

Este capítulo aportará visibilidad a las pruebas de simulación realizadas sobre nuestro sistema. Veremos cada uno de los puntos observados durante la simulación, así como el programa de obtención de errores medios implementado. Este programa se basa en la utilización de diversos algoritmos para el cálculo de la posición del cliente.

Cabe remarcar que en la totalidad de las pruebas de simulación realizadas, nuestro sistema se ha comportado siempre del modo esperado sin ningún tipo de error, una vez finalizado el desarrollo al completo.

Para poder simular nuestro sistema cliente-servidor se ha considerado como mejor herramienta la proporcionada por java, Java Platform Micro Edition Software Development Kit, en su versión 3.0. A su vez también se ha ejecutado en varias ocasiones utilizando Wireless Toolkit en su versión 2.5 para comprobar su correcto funcionamiento en ambos entornos de desarrollo.

5.1 INTERACCIÓN USUARIO-SIMULADOR

Dada la vocación por la sencillez de nuestro programa, el usuario se encuentra ante una aplicación muy intuitiva y de fácil uso. Por ello no existe una gran cantidad de botones, ni elementos seleccionables, ni parámetros que se deban configurar previamente por el usuario.

Desde el punto de vista de la aplicación cliente el usuario deberá, únicamente, pulsar el botón correspondiente para la búsqueda de dispositivos o, por el contrario, salir de la aplicación con el botón “atrás”.

Una vez que los parámetros Bluetooth y de pantalla se hayan inicializado correctamente, se mostrará al usuario que la aplicación se encuentra preparada para obtener la ubicación requerida.

Figura 5.1: Cliente inicio

En el caso de que el sistema encuentre con éxito una localización fiable, la apariencia que muestra el simulador es la que aparece en la figura 5.2. Como puede apreciarse, la interfaz de usuario es sencilla y muestra con claridad la localización calculada.

Figura 5.2: Cliente encuentra localización

Si el usuario quiere manejar el servidor Bluetooth, se encontrará nuevamente con una interfaz sencilla en la que el único elemento que debe manejar es la información de localización. Una vez entra en el modo servidor el programa pedirá que se introduzcan los datos que se desean almacenar con el formato correcto.

Figura 5.3: Insertar localización en servidor

Si el usuario considera que los datos se corresponden con la localización que va a tener el servidor, en el menú podrá encontrar un botón para que dichos datos queden definitivamente almacenados. Una vez hecho esto, el servidor comenzará a correr indefinidamente hasta que el usuario estime necesario pararlo, mediante el botón correspondiente.

En todo momento, el usuario debe tener bien definido un punto de referencia inicial del sistema a partir del cual se dispongan los servidores y el usuario establezca su posición. La ubicación para el punto de referencia elegido, marcará el procedimiento de asignación de posiciones de la totalidad de los elementos presentes en el entorno del sistema.

5.2 PRUEBAS DE SIMULACIÓN

Previo a las pruebas de campo necesarias, utilizando dispositivos reales, se ha considerado oportuno realizar un conjunto de pruebas de simulación para asegurar el correcto funcionamiento futuro del sistema.

En algunas ocasiones, lo que desde un primer momento aparenta ser una aplicación correcta en la simulación, puede causar un sinfín de fallos al portarla a un terminal del mercado con soporte J2ME. Es por esto que se ha acometido un paquete de pruebas consideradas necesarias. Durante todo el proceso de implementación se han realizado las pertinentes comprobaciones para tener constancia de un desarrollo de acuerdo con los objetivos establecidos.

Como prueba más básica, pero necesaria, en todo momento se ha prestado atención al correcto funcionamiento de los botones utilizados y las acciones realizadas una vez presionados. Así, se han realizado diversas comprobaciones en el caso de que el usuario quisiera salir de un modo u otro de funcionamiento mediante el botón “atrás”. Puntos importantes como el cierre de la conexión Bluetooth, parar los hilos o volver correctamente a la pantalla previa han sido comprobados y validados en diversas ocasiones. Para los demás botones utilizados se ha llevado un seguimiento exhaustivo de las acciones realizadas al pulsarse cada uno de ellos, refrendando su funcionalidad.

Una comprobación muy importante, se basa en las búsquedas de dispositivos y servicios realizadas. Es de vital importancia para nuestro sistema que estas búsquedas sean realizadas con normalidad y aporten datos correctos. Por ello se han planteado diversos casos de búsqueda en función de los servidores disponibles. Este banco de pruebas ha sido pasado satisfactoriamente por la simulación de nuestro sistema. Buscar sin servidores, únicamente con un servidor y después eliminarlo, incluir dos servidores para posteriormente quitar o añadir uno más han sido algunas de las pruebas realizadas. En todos los casos de estudio el sistema ha respondido como se esperaba, mostrando la información de error o localización correcta según corresponda.

Otro tipo de pruebas básicas tienen que ver con el formato de los datos de localización introducido. Diversos tipos de formatos han sido introducidos, tanto correctos como erróneos, mostrando nuestro sistema un funcionamiento acorde con lo esperado en todo momento.

Junto con las comprobaciones mencionadas, se ha considerado necesario diseñar una aplicación de simulación de errores medios de los datos de localización reportados por nuestro sistema. Este programa se verá con más detenimiento en el apartado posterior. La idea básica consiste en realizar pruebas automáticamente con un conjunto de datos de localización correctos, para un número determinado de servidores y de esta forma calcular errores medios de precisión, teniendo en cuenta la ubicación real que tendría el cliente (tomada aleatoriamente) y el algoritmo de cálculo de posición utilizado.

Gracias al conjunto de pruebas en simulación realizadas ha sido posible una constante mejora en el funcionamiento del sistema. Por cada comprobación se ha conseguido pulir algo más la presentación final y los resultados obtenidos. De esta forma se ha logrado un sistema robusto con un gran comportamiento frente a errores.

5.3 APLICACIÓN DE SIMULACIÓN DE ERRORES

Para conseguir un banco de pruebas extenso y conocer profundamente nuestro sistema y sus características de funcionamiento, tras un proceso de análisis, se consideró provechoso realizar una aplicación que lograra reportar los errores medios de precisión en los datos de posicionamiento calculados. Esto se consigue comparando la posición que hallaría nuestro sistema con el punto real donde se encontraría el cliente, el cuál se inserta aleatoriamente.

Los errores obtenidos tendrán una ligera variación en función del algoritmo de cálculo de posición utilizado, por lo que se han diseñado y analizado cuatro tipos de ellos para comprobar su precisión.

5.3.1 ALGORITMOS UTILIZADOS

En un primer momento se planteó un único algoritmo de cálculo de posiciones, si bien, para contrastar resultados y obtener una amplia gama de posibilidades se estudiaron tres más que pudieran ofrecer unas prestaciones aceptables. Otros algoritmos analizados no lograban una fiabilidad mínima en los datos reportados (errores medios no superiores a 3.5 metros).

Independientemente del algoritmo utilizado, nuestro sistema será más preciso cuanto mayor sea el número de servidores disponibles capaces de ofrecer información. De este modo la aplicación cliente tendrá un mayor número de datos para poder cruzar y el resultado será más fiable.

El primer, y más lógico, cálculo de posición que consideramos eficaz no es otro que un sencillo medidor de equidistancias. Mediante este algoritmo, el resultado que reporte el cliente en cuanto a su posición será el punto medio respecto a todas las balizas dentro del rango de cobertura. Este algoritmo, al igual que los siguientes, deberá aplicarse para cada una de las coordenadas utilizadas, en nuestro caso, X e Y.

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \qquad \overline{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

Otro algoritmo analizado consiste en añadir pesos al mecanismo de equidistancias, logrando así obtener una media ponderada. Los pesos considerados para el cálculo provienen de la distancia a la baliza más lejana. Este algoritmo se ha incluido para evitar que los servidores más distantes puedan influir en mayor medida que los demás en la solución final. De esta forma, la información que pueda reportar cualquier baliza será considerada de igual modo, evitando así incurrir en errores de servidores con distintas importancias dentro de nuestro sistema.

$$\bar{X} = \sum_{i=1}^{n} \frac{W_i X_i}{W_i}$$, siendo $W_i = d_{near} + 1$

Es conveniente a la hora de calcular los pesos sumar una unidad de distancia, ya que dos balizas pueden tener la misma coordenada X o Y, siendo su distancia cero.

Siguiendo la línea de desarrollo anterior, se ha analizado otro algoritmo con un sistema de pesos diferente. En este caso, se intenta evitar que varias balizas posicionadas a corta distancia unas de otras puedan tirar más hacia sí de la posición final calculada. Es sencillo imaginar que si contamos con diez balizas, de las cuales ocho están en posiciones cercanas y dos más alejadas, con los sistemas anteriormente planteados, el punto final calculado por nuestro sistema, en una recta imaginaria entre un grupo y otro, estaría más cercano del mayor grupo de balizas. Para lograr esto contamos con unos pesos que mostrarán una proporcionalidad inversa al número de coincidencias del valor de la coordenada X o Y con el correspondiente valor cartesiano de las demás balizas dentro del rango de cobertura.

$$\bar{X} = \sum_{i=1}^{n} \frac{W_i X_i}{W_i}$$, siendo $W_i = \frac{1}{n_X + 1}$, con n_X número de balizas con valor de coordenada X idéntico.

Para finalizar, el último algoritmo analizado es una ligera modificación del anterior. En este caso los pesos son calculados del mismo modo, sin embargo en esta ocasión no dividiremos por la suma de los pesos, sino por la suma de las coincidencias encontradas para cada coordenada de las balizas.

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n_x + 1}$$, con n_X número de balizas con valor de coordenada X idéntico.

Por otra parte se podría pensar en una unión entre uno de estos dos tipos de algoritmos y el de la media ponderada. De esta forma conseguiríamos que las balizas ubicadas en una posición lejana, así como las agrupaciones de balizas posicionadas en un espacio reducido no influyeran demasiado en el punto de posicionamiento calculado.

5.3.2 FUNCIONAMIENTO DE LA APLICACIÓN

Con el fin de comprobar la eficacia del sistema de posicionamiento en todos los escenarios posibles, se ha considerado conveniente realizar varias versiones de esta aplicación. Como resultado de esta decisión, podemos pedir al usuario que inserte el número de servidores utilizados o por el contrario insertarlos aleatoriamente.

A su vez, el punto hipotético en el que se encuentra el cliente o la posición de las balizas pueden variar con cada comprobación o mantenerse fijos. Todas estas opciones se han combinado de todos los modos posibles para obtener resultados fiables.

En cuanto a las posiciones de cliente y balizas, se ha considerado adecuado optar por establecerlas aleatoriamente ya que de este modo obtenemos un mayor rango de posibilidades. Por ello, los datos reportados por estas pruebas de simulación en cuanto a errores se refiere, van a ser máximos, no sobrepasando en ningún caso en las pruebas de campo estos errores de precisión.

Debido a estas consideraciones vamos a tratar con mayor detalle las dos versiones de la aplicación cuyas características comunes son:

La posición del cliente varía con cada iteración.
La posición de las balizas se obtiene aleatoriamente.

Ambas versiones difieren en el modo de insertar el número de balizas; por un lado introducidas aleatoriamente, y por el otro pedidas al usuario. Del mismo modo, en el caso aleatorio, independientemente del bucle para calcular medias, se aporta uno más de 20 iteraciones por el que el número de balizas irá variando entre 2 y 29. A su vez, en un caso la posición de las balizas cambiará con cada iteración, mientras que en el otro permanecerá constante, cambiando, únicamente, la posición del usuario. Por tanto, en esencia, estas dos versiones realizadas tienen diferencias al inicio y en la incorporación de un bucle adicional, por lo que se tratarán prácticamente a la par.

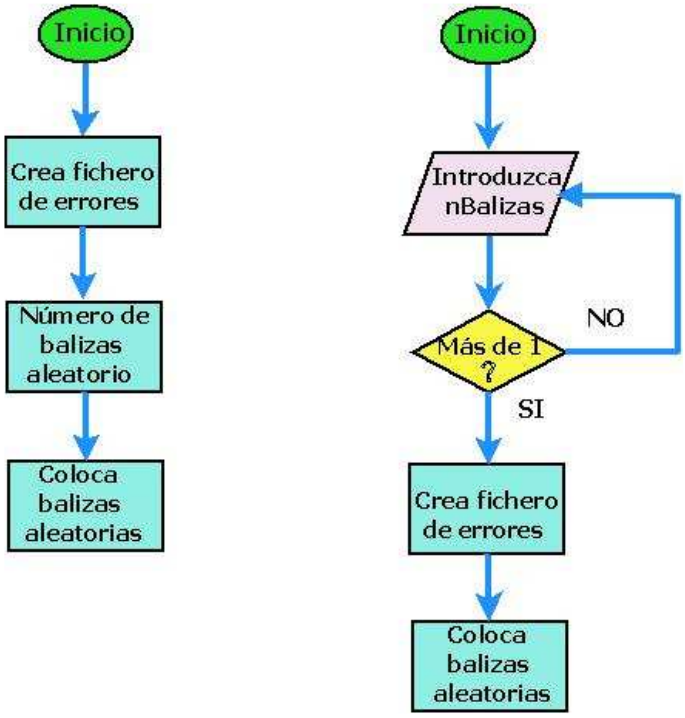

Figura 5.1: Inicio de aplicaciones de simulación

El bucle central de cálculo del algoritmo y sus errores cuenta con 1.000.000.000 de iteraciones, para asegurarnos que los errores de precisión obtenidos son completamente fiables.

Por otra parte, dependiendo de la versión utilizada el usuario podrá interactuar con el sistema, insertando el número de balizas. Esto hace posible que se puedan simular varios escenarios diferentes, desde el más básico y menos fiable, como pueda ser el formado por dos únicos servidores, hasta incrementar la complejidad y fiabilidad lo que el usuario estime conveniente para sus averiguaciones.

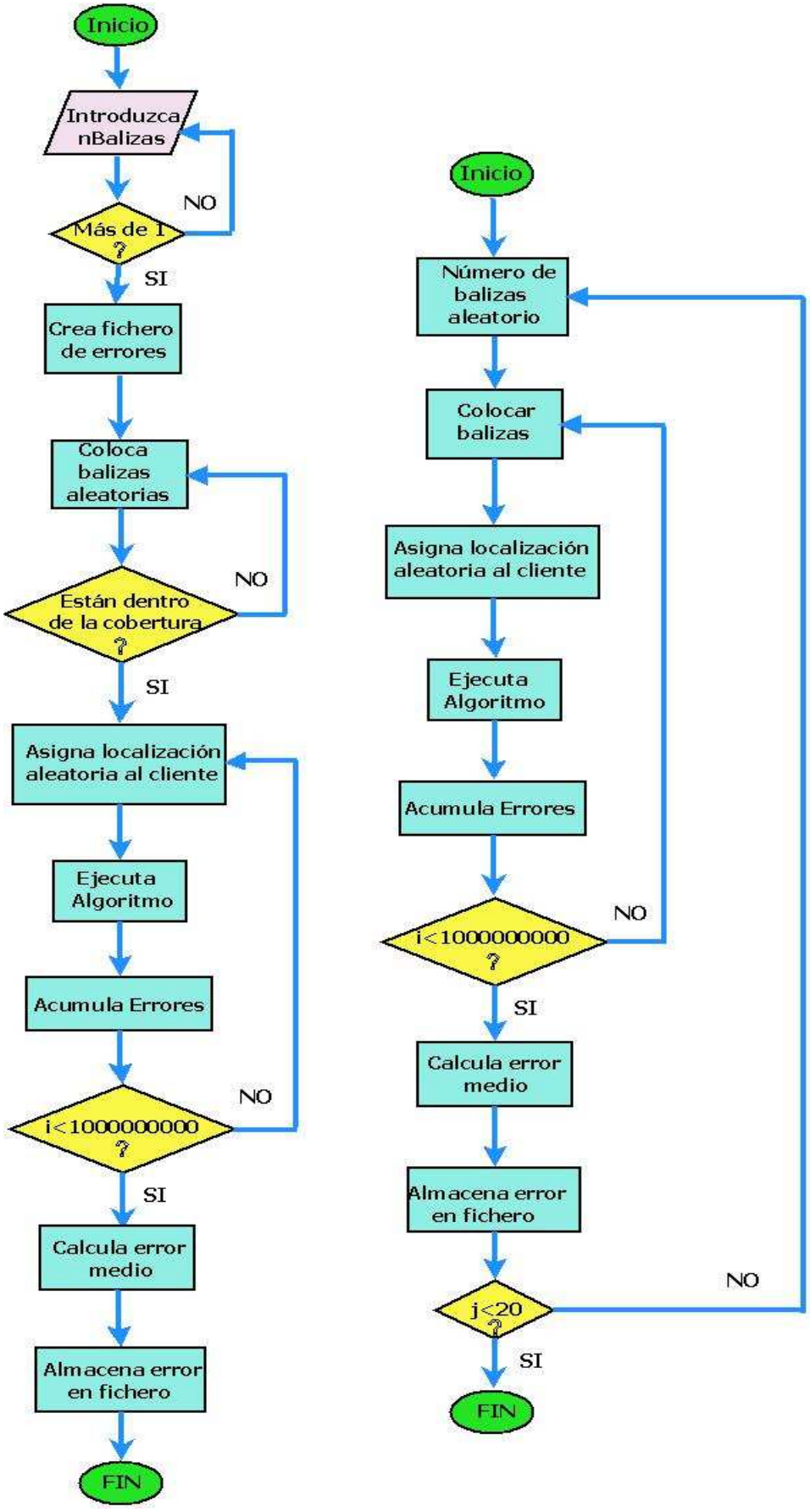

Figura 5.2: Diagrama de flujo versiones de simulador

Los primeros pasos a seguir por ambas versiones muestran algunas coincidencias; creación del fichero al que reportar los datos y elección del número de balizas, para después proceder a su colocación en unos ejes imaginarios de coordenadas cartesianas. Con el fin de conseguir una aplicación capaz de simular al detalle un escenario real de prueba, antes de asignar la posición a cada baliza, se comprueba si ésta se encuentra dentro de la zona de cobertura de la posición que pudiera asignarse al cliente (cobertura óptima Bluetooth V2.1, 10 metros). Por ello seleccionamos la localización de los servidores de entre un rango de coordenadas posibles calculado previamente.

Una vez se han colocado las balizas y la posición del cliente ha quedado fijada, se procede a la ejecución del algoritmo. Como se ha comentado anteriormente, este cálculo se realizará un gran número de veces para comprobar así los errores que incorpora. Para cada iteración de cálculo se almacenarán los errores actuales y una vez terminado se obtendrán automáticamente los errores medios totales. En el caso de la versión aleatoria, estos pasos serán realizados varias veces para asegurarnos unos resultados finales correctos y completamente fiables.

5.3.3 DATOS REPORTADOS

En el apartado anterior, en el que mostrábamos el funcionamiento del simulador de errores, se ha observado que para mantener un seguimiento exhaustivo de los errores de precisión reportados, éstos han ido siendo almacenados constantemente en archivos de texto.

Con todos estos datos obtenidos se han creado tanto tablas, como gráficas descriptivas, apropiadas para una visualización rápida de la precisión lograda utilizando los diferentes tipos de algoritmos comentados anteriormente.

Todos los casos analizados han refrendado nuestras teorías iniciales, cuantos más servidores se encuentren dentro de la zona de cobertura, mejor será la precisión obtenida en el cálculo de la posición del cliente. Como es evidente, al tratarse de un algoritmo ligeramente diferente, en el caso de la versión modificada de las coincidencias entre coordenadas, esta última regla no se cumple. Es por esto que es muy recomendable utilizarlo en casos en los que haya una escasez de servidores, motivo por el cual se implementó. De esta forma tendremos diferentes algoritmos a utilizar dependiendo de los casos de uso posibles

Por otra parte, una vez que el número de balizas aumenta en demasía, el error tiende a equilibrarse. Esto se debe a que el cliente obtiene un aumento de la información redundante. Para evitar datos muy inexactos se ha optado por utilizar un sistema en el que, como mínimo, contemos con tres balizas.

Como es lógico, debido al inmenso número de iteraciones (las máximas permitidas en java por el tipo de dato int), para lograr datos completamente fiables, los errores de las coordenadas X e Y tienden a igualarse cuantas más veces repitamos el algoritmo.

Para asegurarnos unos errores máximos, la posición del usuario, así como la de los servidores varían con cada iteración.

Comenzaremos con el primer algoritmo implementado que calcula una media sencilla, obteniendo un punto equidistante entre todos los servidores.

Balizas	ErrorX	ErrorY
3	5,4985776	5,4986673
4	4,04144464	4,04100244
5	3,9041423	3,90438021
9	3,07536981	3,07523456
10	2,98049414	2,98000061
11	2,90047967	2,90088476
12	2,83699575	2,83687348
13	2,78149281	2,78137344
14	2,73597669	2,7354503
15	2,62359877	2,62654865
20	2,55291314	2,55254393
20	2,55296294	2,5529427
22	2,51380073	2,51363598
23	2,49594744	2,49593538
24	2,48084557	2,48096734
24	2,48088339	2,4808046
27	2,43997636	2,44005508
28	2,42928713	2,42900156
28	2,42926886	2,42918139
29	2,4280749	2,42817438
ERROR MEDIO	**2,909126632**	**2,909182905**

Tabla 5.1. Errores medios algoritmo equidistancias

Atendiendo a estos datos comprobamos que es un algoritmo bastante válido, ya que sus errores de precisión medios son en torno a tres metros, dato completamente aceptable para nuestro tipo de sistema (GPS presenta unas especificaciones en cuanto a errores de hasta 15 metros, estando normalmente en torno a 3,5).

Gráficamente se observa con mayor claridad cómo aumenta la precisión del punto obtenido a medida que el sistema cuenta con más servidores que reporten su posición. Mediante este cálculo podemos observar más claramente como a medida que contamos con más servidores el error disminuye. Este hecho se acentúa aún más si contamos con 3 o 4 balizas únicamente.

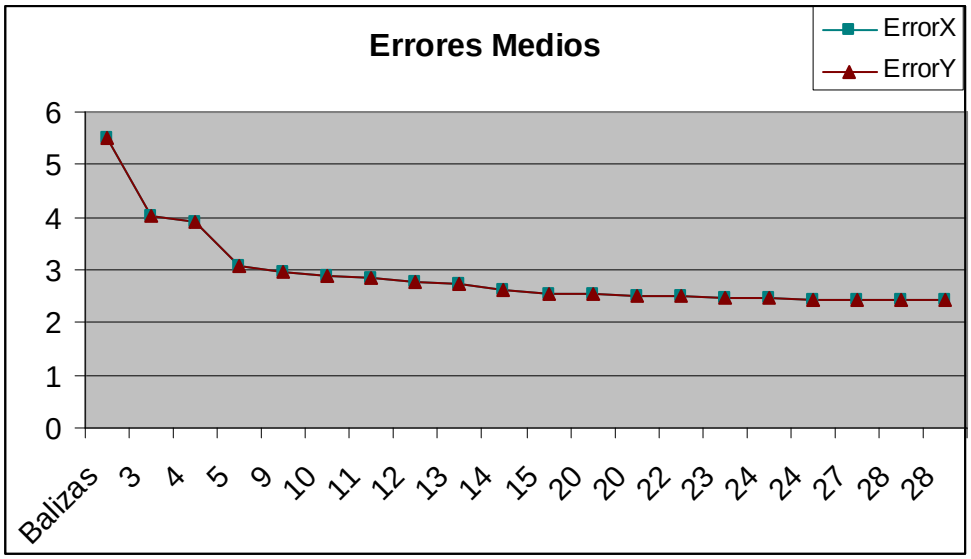

Figura 5.3: Gráfica algoritmo equidistancias

En el siguiente algoritmo a tratar entra en juego un sistema de pesos, para poder hallar una media ponderada. Los datos arrojados por nuestra aplicación simuladora se muestran en la tabla 5.2.

Balizas	ErrorX	ErrorY
3	3,3586	3,1324
5	2,9481	2,9384
5	2,9479	2,9385
6	2,8612	2,8598
7	2,8025	2,8025
8	2,7611	2,7610
10	2,7076	2,7076
14	2,6539	2,6542
15	2,6460	2,6458
17	2,6331	2,6330
19	2,6233	2,6235
20	2,6197	2,6196
20	2,6195	2,6194
21	2,6157	2,6158
22	2,6126	2,6128
23	2,6096	2,6094
23	2,6096	2,6096
25	2,6049	2,6047
26	2,6025	2,6022
28	2,5990	2,5986
ERROR MEDIO	**2,7218**	**2,7094**

Tabla 5.2: Errores medios algoritmo media ponderada

La utilización de este algoritmo aporta un mejor comportamiento frente a situaciones en las que una baliza se encuentre muy alejada. Como en nuestra simulación hemos tomado el rango de cobertura óptimo (diez metros) los servidores no se encontrarán excesivamente alejados. Es por esto que, a simple vista, el comportamiento de este algoritmo no aporte una mejora sustancial.

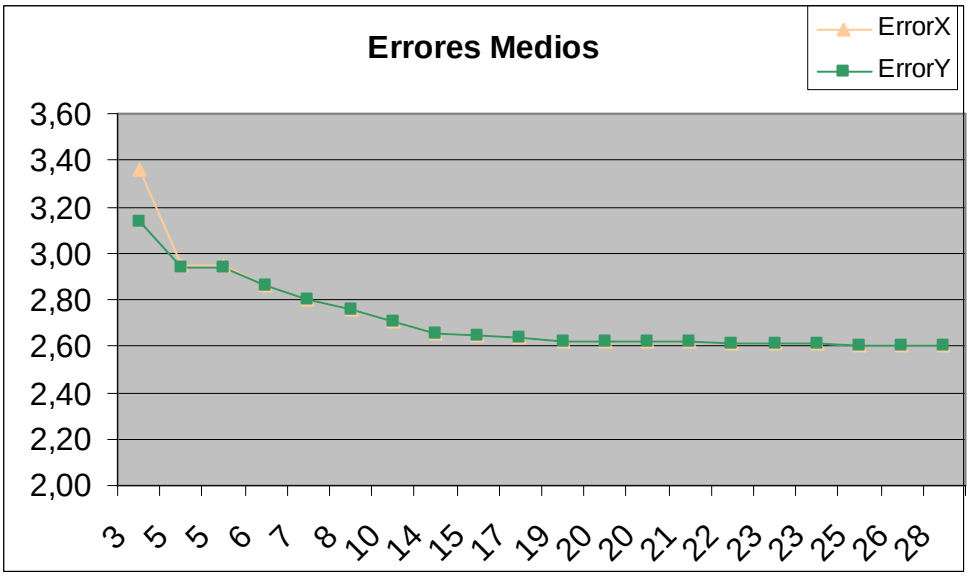

Figura 5.4: Gráfica algoritmo media ponderada

Realizando unos pequeños ajustes en el sistema de pesos obtenemos el siguiente algoritmo a utilizar, llamado de coincidencias. El fin de este algoritmo es evitar que los grupos de servidores en un espacio reducido tiren hacia sí del punto final calculado en mayor medida que los servidores aislados.

Balizas	ErrorX	ErrorY
3	2,95645	2,9871526
4	2,76437839	2,7644154
5	2,65766323	2,65754967
9	2,47430236	2,47434262
10	2,45418825	2,45426602
10	2,45423699	2,45406354
11	2,4387723	2,4386771
12	2,42716206	2,42673425
13	2,41740648	2,41724885
13	2,41708223	2,41740018
14	2,40890656	2,40909635
14	2,40870458	2,40908914
17	2,3906752	2,39103507
21	2,37689904	2,37697332
21	2,37695001	2,37693727

23	2,37321258	2,37341555
24	2,37240298	2,37246155
26	2,37208625	2,37206845
27	2,37291779	2,37261822
29	2,37515285	2,37524173
ERROR MEDIO	**2,438584217**	**2,438612331**

Tabla 5.3: Errores medios algoritmo coincidencias

Demostramos lo apuntado anteriormente, a medida que los servidores aumentan, hay más posibilidades de coincidencias en alguna de sus coordenadas, pero pese a esto no influye para desviar el punto resultante.

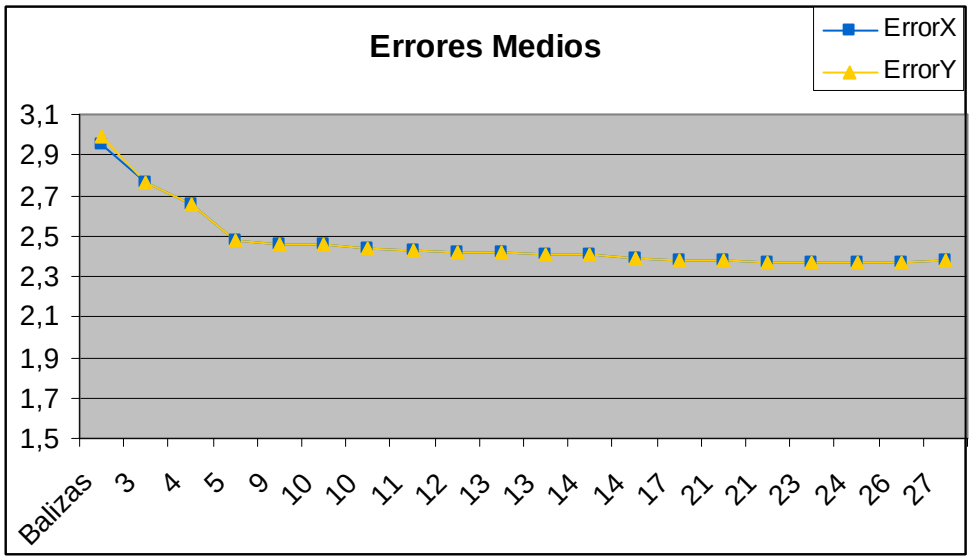

Figura 5.5: Gráfica algoritmo coincidencias

El último algoritmo a tratar se basa en el anterior con una ligera modificación que lo hace un algoritmo un tanto particular. Depende en mayor medida de las coincidencias en las coordenadas de los servidores.

Se ha considerado interesante incluir este algoritmo ya que presenta un comportamiento distinto al resto y puede utilizarse en casos diametralmente opuestos.

Balizas	ErrorX	ErrorY
4	2,41224508	2,41190296
5	2,18168644	2,1815815
5	2,18180563	2,18207484
6	2,03387243	2,03384089

6	2,03382836	2,03362286
8	1,94138237	1,94162417
10	1,94162417	2,04770174
12	2,2469414	2,24730957
14	2,24730957	2,47489754
17	2,80011253	2,80008716
17	2,80010433	2,80014645
22	3,20015327	3,20069877
22	3,2000942	3,20078338
23	3,25901356	3,25933576
23	3,25892986	3,25907999
25	3,3576079	3,35724543
27	3,4380793	3,43893205
28	3,48430912	3,48440813
28	3,48451915	3,48454624
29	3,53585958	3,5365089
ERROR MEDIO	**2,76866798**	**2,768816417**

Tabla 5.4: Errores medios algoritmo coincidencias2

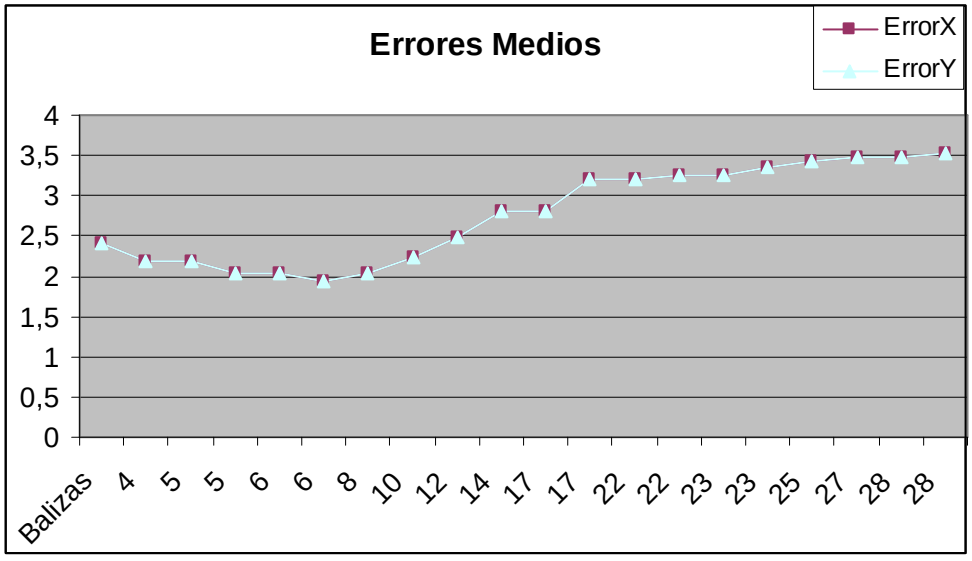

Figura 5.6: Gráfica algoritmo coincidencias2

Gracias a este algoritmo, obtenemos errores medios bastante aceptables, si bien presenta una mayor irregularidad en los datos y a su vez no muestra un comportamiento típico como el esperado en los otros casos. Ya que en este algoritmo las coincidencias entre coordenadas tienen una mayor relevancia, pero puede darse el caso de que no existan, no es capaz de aportar un valor de error medio en disminución a medida que el número de balizas o servidores aumenta.

Con el fin de refrendar lo expuesto anteriormente, vamos a incluir, únicamente, una tabla con los errores medios construida a partir de otros casos de estudio. Como se apuntó anteriormente, en todo momento se ha trabajado en el peor de los casos para obtener errores máximos, esto es, todos los parámetros varían en cada iteración. En esta tabla se muestran los diferentes casos posibles en los que no vamos a encontrar errores máximos.

	Equidistancias		MediaPond		Coincidencias		Coincidencias2	
	ErrorX	**ErrorY**	**ErrorX**	**ErrorY**	**ErrorX**	**ErrorY**	**ErrorX**	**ErrorY**
Punto fijo	1,94	2,10	2,93	2,76	2,40	2,36	2,78	2,31
Balizas fijas	2,84	2,80	2,69	2,71	2,57	2,59	2,62	2,63

Tabla 5.5: Errores medios otros casos

Por otra parte, debido a las múltiples funcionalidades que puede ofrecer nuestro simulador, se ha realizado una gráfica ilustrativa con los errores obtenidos al introducir el propio usuario el número de balizas. Únicamente se incluye el primer algoritmo, equidistancias. Por cada número de balizas se han realizado diez mediciones de errores para, finalmente, realizar el error medio en cada caso.

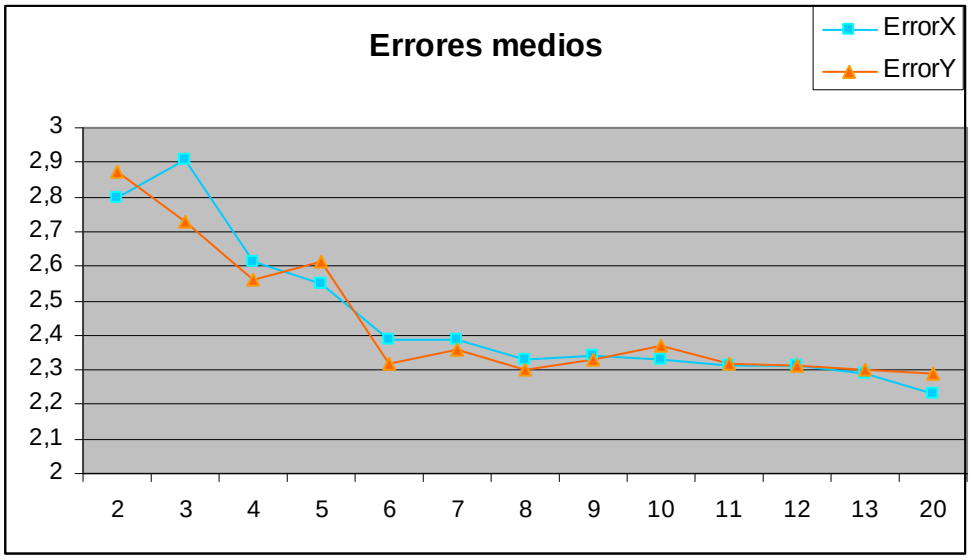

Figura 5.4: Gráfica algoritmo equidistancias (Número de balizas fijo)

6 PRUEBAS DE CAMPO

Una vez que el sistema ha sido implementado en su totalidad y se han realizado las pruebas convenientes, así como el proceso de simulación, es de una importancia capital comprobar el funcionamiento real del sistema en terminales.

Para comprobar su eficacia y fiabilidad se han planteado unos escenarios de distintas exigencias para el sistema. Debido a la variedad de funciones que puede desarrollar, se ha considerado necesario realizar pruebas tanto en entornos de interior como en exteriores teniendo en cuenta en todo momento los rangos de cobertura existentes. Es por esto que, previo al planteamiento de los distintos escenarios se han realizado un gran número de pruebas de cobertura, necesarias para determinar con claridad las limitaciones existentes en los correspondientes entornos de trabajo.

Mediante el conjunto de pruebas de campo que se han acometido, ha sido posible entender y determinar, sin asomo de duda, las prestaciones, errores y limitaciones del sistema creado.

6.1 PRUEBAS DE COBERTURA

Dado el carácter “todoterreno” que se le presupone al sistema, han sido necesarias medidas de cobertura tanto en espacios interiores como exteriores. Además, se han tenido en cuenta diferentes aspectos en cada caso para obtener una extensa descripción de las limitaciones que aporta la cobertura Bluetooth.

En primer término se tuvieron en cuenta los entornos cerrados. La primera prueba básica, que se considera necesaria, consiste en determinar la cobertura máxima obtenida a lo largo de un pasillo diáfano. Los datos arrojados por esta prueba varían en torno a los 17, 18 metros, llegando incluso a distancias cercanas a los 19 metros entre dispositivos. La especificación 2.1 de Bluetooth nos dice que podemos contar con 10 metros de cobertura óptima, por tanto se debe tener en cuenta que a partir de esta distancia estamos fuera de estos parámetros. A su vez, cabe remarcar que se deben considerar las circunstancias de esta primera prueba como “ideales”, sin obstáculos ni interferencias.

Para analizar otro caso de estudio, se ha considerado el hecho de que un dispositivo se pueda encontrar tras una esquina del pasillo. En este escenario, se ha observado la influencia de una nueva variable, la distancia a la pared contraria. Este hecho es lógico debido a que los datos pueden llegar con más facilidad a medida que mi posición va "abriéndose". Para ilustrar con mayor detalle este comportamiento observado se muestra la figura 6.1.

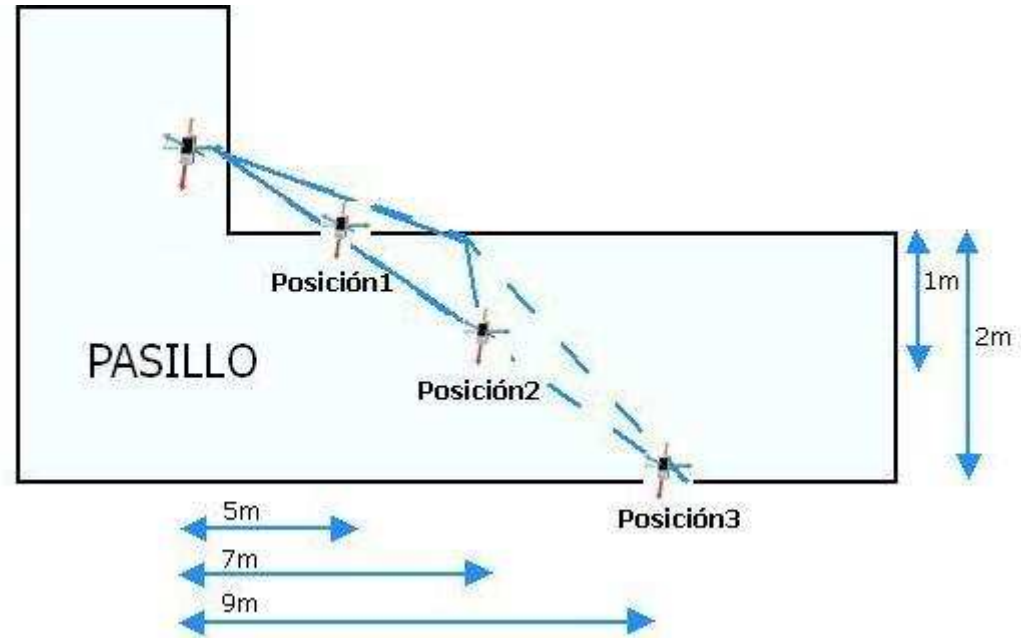

Figura 6.1: Rangos de cobertura en pasillo

Como se puede observar en la posición 1 nos encontramos junto a la pared y la cobertura máxima obtenida es de 5 metros. Si nos alejamos un solo metro, en la posición 2, obtenemos 2 más de cobertura, 7 metros, para, finalmente, en la pared opuesta (2 metros) llegar hasta los 9.

Analizadas estas pruebas básicas, el siguiente paso lógico es comprobar coberturas entre estancias de un lugar cerrado. Para abarcar todas las posibilidades estas pruebas se realizarán con puertas abiertas y cerradas. En el caso de encontrarnos con las puertas abiertas, nos supondrá un menor número de obstáculos por lo que la distancia máxima de cobertura podrá variar entre los 10 y 11 metros. Si por el contrario cerramos las puertas, la distancia a la que los terminales pueden enviarse información disminuye hasta los 8 metros, llegando en algunas ocasiones a los 9.

Del mismo modo se comprobó la cobertura entre plantas de un mismo edificio. Teniendo en cuenta que la altura de la planta es de 2,57 metros y que los terminales se colocaron al borde de la escalera, con una distancia vertical entre ellos de 3, se pudo lograr comunicación hasta con una diferencia de dos pisos, es decir, 6 metros.

Este conjunto de pruebas de interiores son consideradas suficientes para tener una idea general del comportamiento de la tecnología Bluetooth en lo que a coberturas se refiere.

Por otra parte, las pruebas realizadas en lugares abiertos, son más reducidas. Se ha considerado interesante comprobar las distancias máximas de cobertura con diferentes situaciones atmosféricas, si bien los resultados han sido bastante parecidos. Para el caso de una temperatura exterior de unos 16º con viento, las distancias máximas observadas rondan los 10m. Si lo analizamos en circunstancias más estables de 20º con ausencia de viento, podemos llegar en ocasiones a los 11 metros. Además se ha observado otra variable de influencia, de un modo parecido al caso anteriormente mencionado del pasillo. A medida que la altura aumenta podremos obtener una mayor distancia, si bien la proporción no es la misma que en el caso del pasillo ya que por cada metro ganado en altura, únicamente nos podremos distanciar unos centímetros, con un límite entre 11 y 12 metros.

6.2 PRUEBAS EXTERIORES

Al modo del apartado anterior, empezaremos describiendo una prueba básica para, posteriormente, pasar a escenarios más complicados. De esta forma observamos como se comporta el sistema incrementando la dificultad en su ejecución.

El primer escenario básico planteado se desarrolla en un patio con unas dimensiones de 12x7 metros. La sencilla prueba consiste en comprobar si el dispositivo cliente recibe datos de los cuatro dispositivos servidores colocados en cada esquina. Los datos reportados coinciden claramente con los esperados, ya que en todo momento recibe información de las balizas desplegadas. Por tanto, si usamos el algoritmo de equidistancias, el usuario obtendrá la posición media entre los servidores encontrados.

No se ha considerado necesario almacenar datos de la posición obtenida en relación a la real ya que se considera una prueba inicial para analizar el comportamiento en espacios abiertos.

Los otros dos tipos de pruebas pensadas para exteriores son del mismo tipo y se han realizado ya que aportan una característica diferente a la que se puede obtener en espacios cerrados, la distancia. Es por esto que se han utilizado los servidores disponibles para lograr una distancia total considerable. En el primer caso se ha trabajado con 6 dispositivos, alcanzando un espacio de ejecución de unos 50 metros. Para el siguiente escenario se ha necesitado un dispositivo adicional y conseguir de esta forma una longitud total de 60 metros.

En ambos escenarios, debido a la consideración inicial de distancia, sólo se ha tomado en cuenta una coordenada. Esto se debe a que en cualquiera de los escenarios planteados se ha trabajado de acuerdo en todo momento a las limitaciones de recursos existentes, en este caso de terminales. Es por esto que se ha considerado primordial una disposición de los terminales en línea recta para abarcar una mayor distancia.

Todas las pruebas de campo realizadas han sido estudiadas previamente para simular situaciones reales en las que pudiera tener cabida el sistema implementado. Por tanto, gracias a la disposición de terminales planteada para estos escenarios, se consigue emular lo que pudiera ser un pasillo extenso de una galería de arte o un centro comercial.

En estos bancos de pruebas se han analizado otros aspectos, como el comentado en el apartado anterior de rangos de coberturas o el tiempo de reacción de nuestro sistema. Para el caso que nos ocupa, se ha observado un ligero retardo en la obtención de datos de localización por parte del cliente en espacios abiertos. Este es un hecho esperado y aceptable debido a que nos encontraríamos con una calidad inferior del medio de transmisión. Contamos con un tiempo de respuesta del sistema en torno a los 20 o 25 segundos, por tanto el jitter obtenido rondaría los 5 segundos.

Tomadas en cuenta estas consideraciones, la tabla 6.1 muestra los datos recogidos por el sistema en caso de contar con 6 balizas, de 0 a 50 metros. La coordenada que se ha tenido en cuenta es la X, por tanto estos valores muestran las diferencias entre la posición real medida y la reportada por el sistema en este eje. Las balizas han sido colocadas a intervalos de 10 metros para lograr la máxima distancia, ya que como hemos visto, en exteriores, la cobertura Bluetooth no supera esta longitud.

Real	Sistema	Errores
0	0	0
2	0	2
4	5	1
8	10	2
12	15	3
20	20	0
26	30	4
30	30	0
36	35	1
40	40	0
42	40	2
48	45	3
50	50	0
ERROR MEDIO		**1,38461538**

Tabla 6.1: Mediciones con 6 balizas

Como podemos observar el error medio es bastante inferior al obtenido mediante simulación. En todo momento los valores arrojados por el sistema real han sido bastante satisfactorios, si bien, como dijimos en el capítulo de simulación, se habían determinado unos parámetros para obtener errores máximos. Por tanto, salvo algunos valores aislados, alejados de la media, los datos obtenidos están mejorando las expectativas.

Por otra parte, durante el transcurso de estas mediciones, en algunas búsquedas el sistema no fue capaz de encontrar dos servidores válidos debido a la distancia entre balizas en un entorno exterior.

El escenario planteado para el caso en el que contamos con 7 servidores es exactamente igual, con la única salvedad del incremento de la distancia total en 10 metros.

Real	Sistema	Errores
0	0	0
2	5	3
7	10	3
10	10	0
20	20	0
22	25	3
30	30	0
36	40	4
40	40	0
48	50	2
50	50	0
57	55	2
60	60	0
ERROR MEDIO		**1,30769231**

Tabla 6.2: Mediciones con 7 balizas

Es razonable que el error medio varíe, con respecto al caso anterior, algo más de 3 centímetros, es decir, los errores son prácticamente los mismos ya que se trata de un escenario similar.

En ambos casos, si nos posicionamos más allá de la última baliza, obtendremos, como máximo, la posición de ésta hasta que sea posible la conexión con, al menos, dos servidores. Es decir, podremos obtener información hasta un máximo de 62 o 52 metros, dependiendo del escenario. Sobrepasada esta distancia nos aparecerá un mensaje de información insuficiente.

Estas pruebas en escenarios exteriores nos han permitido analizar el funcionamiento del sistema en largas distancias medidas en espacios abiertos. Pese a estas pruebas de “grandes dimensiones” hemos podido comprobar en todo momento el perfecto funcionamiento conjunto de servidores y cliente, reportando unos errores medios de precisión que no superan el metro y medio. Por ello, se puede afirmar con rotundidad que el sistema ha pasado con alta nota este tipo de pruebas planteadas.

6.3 PRUEBAS INTERIORES

El sistema realizado está pensado, sobre todo, para un entorno de ejecución en espacios cerrados, es por esto que se han planteado varios escenarios diferentes para este tipo de pruebas. Para poder determinar las limitaciones imperantes se han realizado las pruebas de cobertura previas. Tomando como base estos datos obtenidos, se han delimitado los diferentes escenarios a tratar.

Los escenarios planteados presentan una complejidad en aumento, desde tres únicos servidores a lo largo de un pasillo, hasta un sistema de balizas dispuestas en varias plantas y habitaciones.

El escenario más básico desplegado consiste en un pasillo interior de unos 30 metros, utilizado a su vez para las anteriores pruebas de cobertura. Contaremos únicamente con 3 balizas y un cliente, por lo que, como se ha venido diciendo a lo largo de esta memoria, el sistema tendrá una menor precisión debido a la poca información que le es enviada al cliente.

Real	Sistema	Errores
0	3	3
2	3	1
3	6	3
5	6	1
6	6	0
7	6	1
8	9	1
10	7	3
12	9	3
14	9	5
16	12	4
ERROR MEDIO		**2,27272727**

Tabla 6.3: Medidas pasillo

Pese a la realización de las pruebas en espacio cerrado, el error de precisión es mayor debido a la escasez de servidores con los que puede interaccionar el cliente. Además la desviación en los datos es mayor, hay errores nulos y errores de hasta 5 metros. Con todo, el error medio continúa siendo aceptable ya que supera escasamente los 2 metros. Por tanto, a la luz de los datos reportados podemos corroborar la fiabilidad del sistema incluso en circunstancias precarias.

Los tiempos de respuesta del sistema también se han considerado importantes en cada uno de los escenarios, debido a que son una característica más del sistema a tener en cuenta, no sólo la precisión. Para este caso, al no contar con demasiadas balizas, estos tiempos son bajos, en torno a los 12, 14 segundos.

En el siguiente escenario planteado es necesaria la incorporación de más servidores, concretamente cinco, uno para cada habitación. Para la organización del escenario se ha tomado la planta de un edificio de unos 60 metros cuadrados. Posteriormente, para comprobar el aumento de precisión, se añadirá una baliza más ubicada en una posición cercana al centro de la planta.

Será descriptivo observar las conexiones que sea capaz de establecer el cliente con los demás servidores, a través de los obstáculos típicos que se pueden encontrar en cualquier vivienda (paredes, puertas, armarios…).

Para la descripción completa del escenario, se incluye la figura 6.2 que muestra la disposición de los servidores normales, así como la del servidor adicional, en recuadro rojo. A su vez, como información complementaria, se incluyen las diferentes medidas de la planta rectangular. Las posiciones en las que se han ubicado los servidores, para todos los escenarios desarrollados, se encuentran adjuntas en el apéndice C. Para el caso actual, la coordenada Z no se ha considerado.

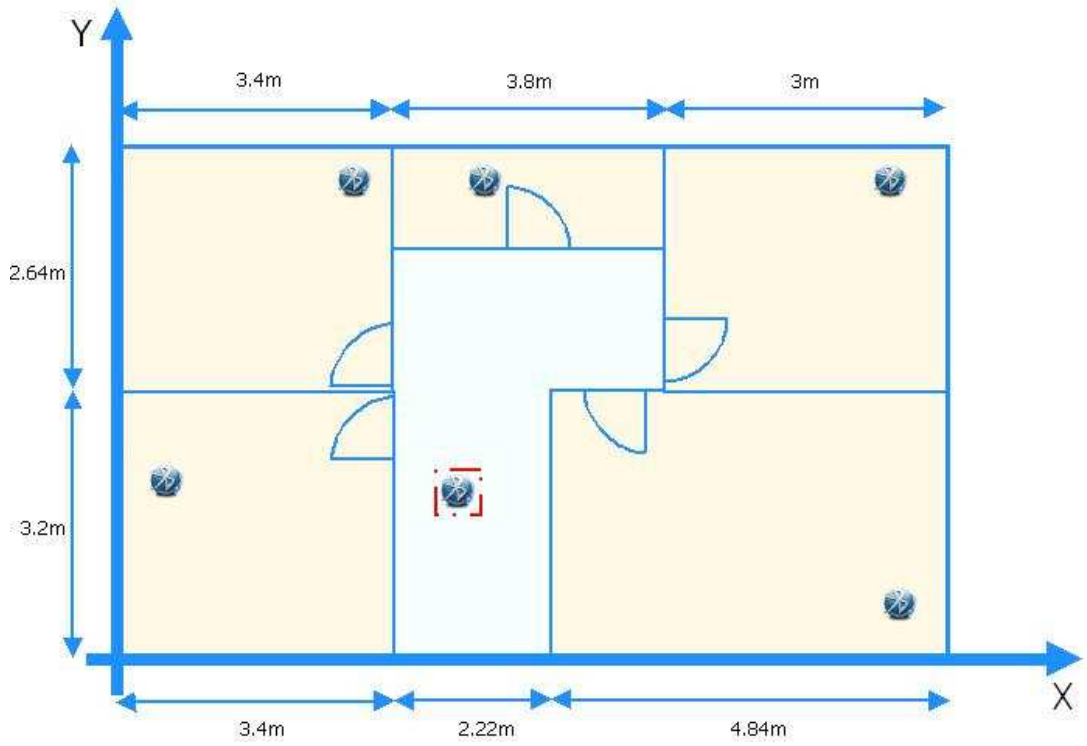

Figura 6.2: Plano de planta de edificio de pruebas

A lo largo de esta prueba el cliente ha ido desplazándose por la totalidad de la planta, para de esta forma comprobar en todo momento la posición mostrada por el sistema en relación a la ubicación real en la que se hallaba.

En la tabla de datos obtenidos, mostrada a continuación, Xr e Yr representan el valor real de la posición, mientras que Xs e Ys el valor reportado por el sistema. Por otra parte, en las dos últimas columnas aparecen representados los errores específicos de cada coordenada.

Xr	Yr	Xs	Ys	ErrorX	ErrorY
5	4	5	5	0	1
1	2	2	3	1	1
1	4	2	3	1	1
5	5	5	5	0	0
9	1	9	4	0	3
9	5	6	3	3	2
ERRORES MEDIOS				**0,83333333**	**1,33333333**

Tabla 6.4: Medidas en planta de edificio

Resulta sencillo comprobar cómo ha aumentado sustancialmente la precisión al contar con cinco servidores susceptibles de aportar información. En algunos casos, dependiendo de la habitación en la que se encuentre el usuario, no se obtendrá información de posicionamiento de la totalidad de las balizas.

Como en casos anteriores, los resultados obtenidos superan con creces las expectativas iniciales y de simulación. Es por esto que el sistema resulta ampliamente eficaz y preciso en este tipo de entornos cuando se cuenta con un mínimo número de balizas.

Los tiempos tomados para este escenario lógicamente aumentan, fluctuando entre los 15 y 18 segundos. Cabe remarcar que en ciertas mediciones el sistema es capaz de ofrecer un tiempo de menos de 10 segundos, valor que se considera realmente bajo.

El siguiente conjunto de medidas presenta una ligera modificación que pretende incrementar, si cabe, la precisión obtenida. Vamos a trabajar con seis servidores, es decir, incluimos a las balizas dispuestas por todas las habitaciones, el servidor adicional que veíamos recuadrado en rojo en la figura 6.2. Con ello esperamos un incremento en la información recibida y por tanto, unos datos finales más completos y precisos.

Xr	Yr	Xs	Ys	ErrorX	ErrorY
5	4	5	4	0	0
1	2	2	2	1	0
2	4	2	3	0	1
5	5	5	4	0	1
9	1	7	3	2	2
9	5	9	5	0	0
ERRORES MEDIOS				**0,5**	**0,66666667**

Tabla 6. 5: Medidas en planta de edificio con servidor adicional

Acorde con el comportamiento esperado, la precisión de los datos arrojados por el sistema se ha incrementado, llegando en media hasta casi medio metro. Es por esto que el segundo escenario planteado de este tipo presenta un mejor comportamiento, por lo que se aconseja la utilización del servidor adicional. Incrementar en demasía el número de balizas aportaría, posiblemente, información redundante por lo que, hablando en términos apriorísticos, podría no ser necesaria su inclusión en el sistema.

El efecto positivo que supone esta inclusión, no se ve igualmente reflejado en el caso del tiempo de respuesta. Como cabe esperar, aumenta mínimamente, en torno a 2 segundos, por tanto nos iríamos hasta un tiempo total de 20 o 21 segundos. Si bien, en la mayoría de las mediciones, este periodo no ha variado apreciablemente.

Para conseguir un análisis más exhaustivo de la funcionalidad del sistema en interiores, se ha considerado de gran utilidad incluir un factor más al escenario anteriormente descrito. Para esta prueba, las puertas del edificio permanecerán cerradas en todo momento. Con ello se pretende comprobar si la respuesta del sistema sigue siendo aceptable incluso en casos complicados. En esta primera toma de medidas no se incluirá la baliza adicional.

Xr	Yr	Xs	Ys	ErrorX	ErrorY
5	4	5	3	0	1
1	2	2	3	1	1
1	4	3	4	2	0
5	5	4	4	1	1
9	1	7	3	2	2
9	5	6	4	3	1
ERRORES MEDIOS				**1,5**	**1**

Tabla 6.6: Medidas en planta de edificio con puertas cerradas

Los datos alcanzados por el sistema, como era de esperar, presentan un aumento en el error medio. Este hecho concuerda con los valores esperados, dado el caso de que se le han interpuesto al cliente más obstáculos para su comunicación con los servidores. Pese a esto, comprobamos que los errores de precisión calculados continúan siendo bastante buenos. Si a este escenario le añadimos el servidor adicional o central, los valores reportados cambiarán.

Xr	Yr	Xs	Ys	ErrorX	ErrorY
5	4	5	3	0	1
1	2	3	3	2	1
2	4	3	3	1	1
5	5	3	4	2	1
9	1	7	2	2	1
9	5	6	5	3	0
ERRORES MEDIOS				1,66666667	0,83333333

Tabla 6.7: Medidas en planta de edificio con puertas cerradas y servidor adicional.

A la luz de los datos obtenidos, comprobamos que la mejora aportada por el servidor adicional no es tanta como en los escenarios anteriores. El factor de puertas cerradas mitiga el efecto positivo que puede causar dicho servidor. Esto es lógico, ya que en determinados puntos del edificio, el cliente no podrá lograr una conexión de garantías con esta baliza pese a tener una ubicación casi central.

Desde el punto de vista de tiempo de respuesta, al presentar una mayor dificultad en las conexiones, el sistema ofrece un incremento poco sustancial, en torno a los 23 segundos.

Para terminar con las pruebas planteadas, se ha optado por incrementar un nivel más la dificultad del escenario, por ello la exigencia del sistema será mayor. Con estas últimas mediciones se considera cerrado el apartado de pruebas, ya que se ha sometido constantemente al sistema a varias situaciones de diferente dificultad y exigencia. Por tanto estaríamos en posición de determinar, sin lugar a dudas, las prestaciones que, en todo momento y situación, podría ofrecer así como sus requisitos de ejecución.

En el escenario final toma relevancia una tercera coordenada. Esto se debe a que las balizas se dispondrán en diferentes pisos de un edificio, en concreto tres. El usuario se desplazará por entre estas tres plantas obteniendo en cada momento su ubicación. Para ello se han dispuesto tres balizas por planta, a excepción del último piso, en el que únicamente se ha colocado una baliza por falta de recursos. Para lograr una buena disposición de los servidores se ha considerado conveniente colocar uno en el centro y dos en los laterales de cada planta.

Xr	Yr	Zr	Xs	Ys	Zs	ErrorX	ErrorY	ErrorZ
4	4	1	3	2	1	1	2	0
1	4	1	3	3	1	2	1	0
2	4	1	4	2	1	2	2	0
4	5	1	4	3	1	0	2	0
4	2	1	4	3	1	0	1	0
8	4	1	5	3	1	3	1	0
4	2	0	4	3	0	0	1	0
4	1	0	3	3	0	1	2	0
1	1	0	2	3	0	1	2	0
2	4	0	2	4	0	0	0	0
4	4	0	4	2	0	0	2	0
3	5	0	3	3	0	0	2	0
8	4	0	6	3	0	2	1	0
4	1	1	4	3	1	0	2	0
2	1	1	3	2	1	1	1	0
ERRORES MEDIOS						**0,86666667**	**1,46666667**	**0**

Tabla 6.8: Medidas en varias plantas de edificio

Pese a ser un caso de cierta exigencia para nuestro sistema, vuelve a demostrar un gran comportamiento, rebajando significativamente en cualquier prueba realizada los valores obtenidos en simulación. Por tanto hemos comprobado que se puede añadir un nuevo entorno de ejecución para el que es válido el sistema, ubicación de un usuario entre plantas de un edificio.

Al no contar con balizas suficientes en el segundo piso, no se han realizado varias mediciones. Únicamente una, para observar el valor de la coordenada Z obtenido. Como era de esperar, corresponde con el piso en el que se encontraba el cliente.

Analizando minuciosamente la totalidad de los datos obtenidos durante todo el periodo de pruebas, podemos afirmar que el sistema tiene un perfecto comportamiento en cualquier entorno de ejecución y presenta unos errores de precisión ínfimos para la funcionalidad que aporta. Además, los tiempos de ejecución son perfectamente asumibles por cualquier usuario que utilice el sistema.

Para aportar una visión global de las prestaciones y requisitos del sistema, se muestra esta tabla resumen en la que aparecen los niveles reportados para cada uno de los parámetros analizados durante todo el periodo de pruebas. Con ella se pretende mostrar, en una sola visión, las características del sistema completo.

Escenarios	Tiempo respuesta	Precisión	Gasto recursos
Planta(sin adicional)			
Planta(con adicional)			
Pasillo			
Entre plantas			

Tabla 6.9: Resumen general del comportamiento del sistema

7 PRESUPUESTO

A lo largo de este capítulo se expondrá, detalladamente, el ámbito monetario del proyecto realizado. Se pretende adoptar un baremo completamente fiable y acorde con la realidad de nuestros días, por lo que cada uno de los aspectos aquí tratados se ajustarán en la medida de lo posible a los valores manejados en el mundo laboral.

Comenzaremos tratando los recursos más costosos, los personales. Para aportar veracidad a nuestros datos, consultamos al colegio oficial de ingenieros técnicos de telecomunicaciones (COITT). Ocurre que se ha suprimido la función de los Colegios de fijar baremos orientativos de honorarios o cualquier otra recomendación sobre precios ya que provocan una restricción de forma injustificada[11]. Debido a esto se ha estimado conveniente adoptar los últimos baremos conocidos del COITT. En el caso que nos ocupa, un ingeniero técnico recién titulado viene a percibir unos 21.000€ brutos al año, contando esto en 14 pagas. A la vista de estos datos, el sueldo medio rondaría los 1500€/mes. Si bien, a efectos de la empresa, mantener a un trabajador supone, más o menos, el 1,5 de la nómina mensual si tenemos en cuenta aspectos como la seguridad social, por tanto tendríamos:

14 pagas * 1.500€/mes * 1,5 = 31.500€/año

Si a esto le unimos que un trabajador ronda las 1.575 horas al año, descontando días no laborables, vacaciones…

31.500€/año / 1750horas/año = 18€/hora

Este sería el importe que debería abonar la empresa por cada hora trabajada de un ingeniero técnico.

El diseño de este sistema discurre desde el mes de Marzo de 2010 hasta el mes de Junio de 2010. Si aproximamos las horas de trabajo a un total de 4 horas por la mañana y 3 horas por la tarde, obtenemos un total de unas 7 horas diarias. Este dato ha podido variar cada día, si bien es una media de tiempo trabajado bastante fiable.

7horas/día * 20 días/mes * 4meses = 560 horas

El tiempo de trabajo, aunque no se han contado las horas empleadas durante los fines de semana, se ajusta bastante a la realidad. Como en cualquier calendario de trabajo se ha contado con 20 días trabajados al mes.

Concepto	Horas	Honorarios	Importe
Ingeniero Técnico de Telecomunicaciones	560 horas	18€/hora	10.080€

Tabla 7.1: Coste personal

En lo que al coste material se refiere, destacaremos que no es necesario comprar licencias de ningún tipo, ya que se ha realizado con software libre. Aparte del PC de sobremesa utilizado, se han necesitado 10 terminales móviles para las pruebas de campo. No se ha contado con modelos iguales, por tanto se supone un valor medio de 100€ por terminal. Si tenemos en cuenta una vida útil de cada móvil de 2 años, debido a las condiciones de permanencia y demás ofertas, y que la utilización de los mismos ha sido durante 3 meses, eliminamos el valor residual. Ocurre del mismo modo con el PC si le suponemos una vida de 5 años, siendo la utilización de 6 meses.

100€ * 3 meses / 24 meses = 12,5€
800€ * 6 meses / 60 meses = 80€

Concepto	Unidades	Precio (unidad)	Amortización	Importe
Pc sobremesa	1	800€	80€	80€
Terminal móvil	10	100€	12,5€	125€
TOTAL				**205€**

Tabla 7.2: Coste material

Otro gasto a tener en cuenta es el material fungible, es decir, que no es inventariable y se gasta a medida que transcurre el proyecto.

Concepto	Importe
Folios, post-it…	6€
Cartuchos impresora, CD´s	20€
Bolígrafos, clips, tippex…	10€
TOTAL	**36€**

Tabla 7.3: Coste material fungible

Por último, al subtotal del presupuesto se le aplicará un 3% adicional en concepto de costes indirectos, que pueden derivar, por ejemplo, de los gastos de administración.

Concepto	Importe
Coste de personal	10.080€
Coste de material	205€
Coste bienes fungibles	36€
Costes indirectos (3%)	357,48€
Subtotal (Sin IVA)	10.678,48€
IVA (16%)	1.708,55€
TOTAL	**12.387,03€**

Tabla 7.4: Costes totales

8 CONCLUSIONES

Este proyecto nació, como se expone en la introducción, con la intención de "unir" dos de las tecnologías mas extendidas en la actualidad, el sistema de posicionamiento global, GPS y la tecnología de comunicación inalámbrica más utilizada en dispositivos móviles, Bluetooth. Desde un primer momento se trabajó con ahínco para lograr este objetivo inicialmente planteado. El resultado final debía ofrecer unas prestaciones similares a GPS en cuanto a posicionamiento, utilizando únicamente la conexión inalámbrica entre un número limitado de dispositivos. Toda vez que el sistema implementado se basa en dicha tecnología, se ha trabajado en todo momento con las limitaciones impuestas por la misma, poniendo especial atención en los rangos de cobertura óptimos.

A modo de premisas iniciales, el desarrollo del sistema se encauzó hacía unas líneas de diseño basadas en el ahorro de recursos, tan escasos en los terminales móviles, en la sencillez y eficiencia de ejecución y, sobre todo, en la búsqueda de novedades y soluciones alternativas a las ya existentes en este tipo de aplicaciones. Estos puntos de acción han dado como resultado un sistema original en su diseño, simple en su ejecución y sencillo en lo que a experiencia de usuario se refiere, sin que por ello se hayan descuidado términos como la precisión y fiabilidad del mismo.

Fruto del gran peso que la investigación ha supuesto para este proyecto, se han podido satisfacer las necesidades iniciales e, incluso, superarlas en las pruebas posteriores. El análisis y estudio de todos los recursos a nuestro alcance, han ofrecido unos resultados finales en los que los atributos de servicio (DataElement) han supuesto la principal novedad del sistema y una importante innovación con respecto a los otros sistemas existentes implementados con JSR82. Es por esto que podemos afirmar, sin lugar a dudas, que el resultado conseguido supone un avance en cuanto a ahorro de memoria, en la medida en que se han podido omitir fragmentos de código considerados "necesarios" en este tipo de aplicaciones, eficiencia del procesador, no es necesaria la ejecución de tantos comandos, o velocidad, ya que se ha conseguido reducir al mínimo el intercambio de información y los comandos necesarios para llevarla a cabo. Además, otro importante hito conseguido es la duración de la conexión entre cliente y servidor, ésta se puede interrumpir una vez se ha obtenido la información deseada con el consiguiente ahorro de baterías que ello supone.

Por todo ello se ha logrado desarrollar un sistema de posicionamiento fiable, muy preciso, con una funcionalidad variada y distintos entornos de ejecución posibles. En esencia puede llegar a aportar unos resultados similares a GPS para un entorno reducido, si bien, debido a su diseño presenta algunas mejoras respecto a éste. Al no tener la necesidad de una conexión vía satélite, nuestro sistema es capaz de ofrecer información de localización inmediatamente después de ser ejecutado. El tiempo de arranque se debe a la inicialización de los parámetros Bluetooth necesarios. El consumo de baterías es bastante menor ya que no es necesario triangular la información de localización de al menos tres satélites, sino de varios servidores dispuestos a una corta distancia. Por último las especificaciones de GPS confieren a este sistema una precisión de 15 metros, si bien, normalmente el error rondará los 3[12], por lo que a la vista de los resultados de las diferentes pruebas aportadas, el sistema implementado muestra una mayor precisión en su entorno de ejecución. En ambos sistemas la precisión dependerá del número de dispositivos o satélites visibles en ese momento.

Como factor independiente del sistema, pero inherente a la obtención de resultados satisfactorios, cabe remarcar la buena coordinación, predisposición y constante comunicación entre proyectando y tutor para la consecución de este proyecto. Sin esta fluida interacción se degradarían los resultados obtenidos y, sobre todo, el tiempo necesario para la realización del proyecto en su totalidad.

Teniendo en cuenta todas las consideraciones descritas, el sistema final supera con creces las expectativas iniciales respecto a los hitos que se pretendían alcanzar, suponiendo una especial relevancia los ínfimos errores de precisión que presenta en los diferentes entornos de prueba y simulación planteados. Por tanto, se trata de un sistema que proporciona un gran comportamiento al usuario y se posiciona como un gran punto de partida para una interesante línea de evolución futura.

9 POSIBLES MEJORAS Y AMPLIACIONES

La aplicación realizada debido a su carácter abierto puede contemplar innumerables ampliaciones y mejoras en lo que se refiere a la interfaz, al funcionamiento o a los valores que reporta. Pasamos a analizar algunas de las más interesantes con sus características principales.

9.1 INTERFAZ GRÁFICA

Comenzaremos definiendo unas posibles mejoras en cuanto a la interfaz gráfica se refiere. Estas ampliaciones podrían ser válidas para una comercialización más intuitiva y práctica, encaminada a un usuario medio, acostumbrado a unas aplicaciones de uso bastante sencillo.

Una primera aproximación sería incluir imágenes en las listas que aparecen en la aplicación o insertar más alarmas de tipo informativo para mejorar la apariencia del programa. De esta forma el código origen no se modificaría demasiado y la aplicación tendría una apariencia con mayor presencia gráfica. Para el caso que nos ocupa, no hemos considerado primordial añadir estos elementos.

Otra posible solución podría ser trabajar con una interfaz de usuario de bajo nivel, utilizando las clases *Canvas* y *Graphics* para obtener el manejo gráfico de la pantalla a nivel de píxel. Con ello obtendríamos un mayor control sobre lo que nuestro programa mostrará por pantalla y una mayor flexibilidad, si bien, se reducirá considerablemente la portabilidad de nuestra aplicación y el resultado final dependerá del teléfono móvil en el que esté ejecutándose. Esto se debe a que necesitamos obtener el tamaño del display y a partir de éste programar nuestra interfaz gráfica.

Como una posible mejora más encaminada a una interfaz gráfica con apariencia de juego, podríamos utilizar la librería JSR184 para modelar con tres dimensiones. Esta es, posiblemente, la ampliación menos eficiente del programa debido a la introducción de un API algo complicado, engorroso y poco conocido. Además, no es de vital importancia mostrar la información obtenida en un formato de 3D.

Una vez vistas y analizadas las anteriores soluciones y tras un proceso de búsqueda y documentación llegamos a la posible mejora más conveniente para nuestra aplicación. Se trata de LWUIT, una librería gráfica fácil de comprender y con bastantes similitudes con Swing o AWT.

Estas son algunas de sus características más importantes:

- Amplía la funcionalidad de `javax.microedition.lcdui`.
- Aporta controles gráficos como `ComboBox`, `ListBox` o `TabbedPane`.
- Animaciones en 2D/3D.
- Incorpora casi todos los `Layouts` definidos en AWT.
- Eventos y listeners.
- Licencia de uso libre.

Junto a estas características encontramos otra que, actualmente, puede tener una mayor relevancia. LWUIT soporta dispositivos táctiles. Esto es, sin duda, algo remarcable debido al auge de los móviles touchscreen y su continua expansión y evolución en el mercado.

Figura 9.1: Ejemplo LWUIT

Como se puede observar en la figura anterior, la apariencia obtenida es bastante similar a la que se puede alcanzar con AWT o Swing. La cantidad de posibilidades de formato de pantalla es bastante extensa por lo que se considera una mejora muy interesante.

9.2 FUNCIONAMIENTO

En lo que a la mecánica de la aplicación se refiere, también podemos incluir varias actualizaciones. De entre todas ellas, vamos a destacar las que consideramos más importantes o aprovechables.

Suponiendo un entorno con varios clientes que requieran su localización, podría establecerse un sistema de colas en cada uno de los servidores-baliza. De esta forma aseguramos que todos los clientes puedan obtener su localización en un reducido espacio de tiempo, dependiendo del número de clientes existentes en la zona de cobertura. A priori, la implementación no requeriría de una gran cantidad de código adicional, siendo necesario para lograrla un Vector que emule la cola en el servidor y un hilo adicional que vaya manejando las conexiones de los clientes constantemente hasta que el servidor se parara.

Dentro de este mismo caso de mejora, podríamos ir más allá y crear un sistema de prioridades en vez de una simple cola o pila. Cada cliente tiene un identificativo según su prioridad y si éste se corresponde con una prioridad alta el servicio sería dado automáticamente a dicho cliente por encima de cualquier otro con menor prioridad. Esto se podría conseguir estableciendo, por ejemplo, un atributo adicional del mismo servicio en el que se definiera el tipo de prioridad. Así, dependiendo del tipo de cliente que sea tendrá su correspondiente valor del atributo en cuanto a la prioridad se refiere.

En nuestro entorno de ejecución no suponemos un alto número de clientes como para incluir esta mejora, por ello hemos preferido incluirla en este apartado en vez de implementarlo.

Como otra posible mejora, sería provechoso almacenar un histórico de las posiciones que nuestra aplicación nos ha ido calculando. Esta ampliación podría servirnos para utilizar nuestra aplicación en determinadas situaciones, como puede ser la visita a un museo o un centro comercial. De esta forma ninguna sala de la galería de arte o zona del centro nos quedaría por visitar.

En principio, nuestro programa deja fijas en pantalla las sucesivas posiciones para poder ir comprobándolas, luego nos encontramos en un paso intermedio hacia esta mejora. Únicamente habría que almacenar esta información, mostrándola cuando el usuario así lo requiriera.

Por último, a modo de un sistema de servidores, el cliente podría almacenar una "lista negra" con los servidores que envían constantemente, o más a menudo, información errónea. De esta forma, cada vez que llega información, es comprobada su procedencia, mediante su dirección Bluetooth y es tomada en cuenta o no.

Con ello se podrían evitar varias comprobaciones en el formato de la información por parte del cliente obteniéndose así una información más fiable.

9.3 VALORES REPORTADOS

Nuestra aplicación muestra valores de posicionamiento tomando cualquier punto de referencia que nosotros estimemos. Para mejorar la precisión de los datos y estandarizarlos, se podría pedir al iniciar el servidor y el cliente el tipo de datos que se quiere reportar. Coordenadas cartesianas o coordenadas geográficas serían un buen ejemplo.

Por otra parte junto con los valores de posición mencionados, una mejora que aportaría mucha funcionalidad y versatilidad a nuestro sistema podría ser incluir información adicional. Estos datos adicionales podrían ir desde una lista con los puntos de interés más cercanos hasta la distancia a la salida más próxima pasando por la dirección hacía la que tengo que dirigirme para encontrar mi vehículo en el parking o dirección de cuadros o tiendas más visitados, dependiendo del entorno de ejecución (museos, centros comerciales, aparcamientos…). Esta mejora ralentizaría ligeramente nuestro sistema por lo que al inicio se podría preguntar al usuario por dos modos de funcionamiento, normal o con información adicional.

Para lograr implementar esta mejora el principal cambio a efectuar sería en el cliente. Toda la información posible que se deseara incluir habría que cargarla previamente para, en función de la posición obtenida, mostrar los datos adicionales.

10 DIAGRAMA DE GANTT

La consecución de cualquier tipo de proyecto que se precie, debe llevar acarreada una correcta planificación del desarrollo del mismo. Si cabe, es aún más necesario en nuestro caso, dado el escaso tiempo con el que se contaba desde un primer momento. Por ello ha resultado primordial para el correcto cumplimiento de los plazos inicialmente establecidos, una correcta planificación junto con un trabajo constante. Los periodos de tiempo marcados como premisas a cumplir inicialmente, han sufrido variaciones prácticamente imperceptibles.

Como es de esperar, el desarrollo del proyecto se ha dividido en varias etapas o tareas generales, dispuestas por orden de inicio que engloban a otras tareas más específicas.

Documentación: Es requisito indispensable e ineludible el estudio, análisis y comprensión de todas las herramientas disponibles para la realización del proyecto.

Implementación del cliente: Se comenzó desarrollando la aplicación cliente que ofrece al usuario su localización.

Implementación del servidor: Una vez terminado el cliente, se realizó la implementación del servidor que ofrece los servicios al cliente.

Pruebas cliente-servidor: Para probar la eficacia del sistema bajo cualquier exigencia, han sido necesarias varias pruebas en el entorno de simulación.

Implementación de simulador: Con el fin de probar nuevos algoritmos y comprobar errores de precisión.

Pruebas de campo: El sistema implementado ha sido pensado para ejecutarse en dispositivos móviles por ello han sido necesarios varios días de pruebas “reales”.

A continuación se muestra en la figura 10.1 una descripción general de las fases del proyecto junto con sus tareas específicas, así como las fechas de inicio, fin y duración de cada una. Del mismo modo, se pueden observar las tareas vinculadas mediante los predecesores.

	Nombre	Duración	Inicio	Terminado	Predecesores
1	**Sistema de posicionamiento**	113 days	**5/03/10 8:00**	**25/06/10 17:00**	
2	**Documentación**	21 days	**5/03/10 8:00**	**25/03/10 17:00**	
3	Estudio Bluetooth	8 days	5/03/10 8:00	12/03/10 17:00	
4	JSR82	10 days	13/03/10 8:00	22/03/10 17:00	3
5	MIDP	1 day	23/03/10 8:00	23/03/10 17:00	4
6	Otras tecnologías	2 days	24/03/10 8:00	25/03/10 17:00	5
7	**Implementación Cliente**	41 days	**26/03/10 8:00**	**5/05/10 17:00**	**2**
8	Creación de pantalla	2 days	26/03/10 8:00	27/03/10 17:00	
9	Estudio de soluciones alternativas	2 days	28/03/10 8:00	29/03/10 17:00	8
10	Creación de conexión alternativa	4 days	30/03/10 8:00	2/04/10 17:00	9
11	Cálculo de posición final	3 days	3/04/10 8:00	5/04/10 17:00	9;10
12	Funcionalidad total	27 days	6/04/10 8:00	2/05/10 17:00	11
13	Corrección de errores y pruebas aisladas	3 days	3/05/10 8:00	5/05/10 17:00	12
14	**Implementación Servidor**	27 days	**6/05/10 8:00**	**1/06/10 17:00**	**2;7**
15	Creacion de pantalla	1 day	6/05/10 8:00	6/05/10 17:00	
16	Creación de conexión alternativa	2 days	7/05/10 8:00	8/05/10 17:00	15;9
17	Correcciones de formato	1 day	9/05/10 8:00	9/05/10 17:00	16
18	Funcionalidad total	20 days	10/05/10 8:00	29/05/10 17:00	17
19	Corrección de errores y pruebas aisladas	3 days	30/05/10 8:00	1/06/10 17:00	18
20	**Pruebas Cliente-Servidor**	5 days	**2/06/10 8:00**	**6/06/10 17:00**	**14;7**
21	Pruebas intercambio de datos	4 days	2/06/10 8:00	5/06/10 17:00	18;12
22	Pruebas funcionalidad completa	4 days	3/06/10 8:00	6/06/10 17:00	12;18
23	**Implementación Simulador**	20 days	**6/06/10 8:00**	**25/06/10 17:00**	
24	Análisis de algoritmos	3 days	6/06/10 8:00	8/06/10 17:00	
25	Estudio de posibles funciones	1 day	9/06/10 8:00	9/06/10 17:00	24
26	Funcionalidad completa y versiones	9 days	10/06/10 8:00	18/06/10 17:00	25
27	Pruebas y almacenamiento	6 days	19/06/10 8:00	24/06/10 17:00	26
28	Desarrollo de gráficos y tablas	1 day	25/06/10 8:00	25/06/10 17:00	
29	**Pruebas de campo**	6 days	**19/06/10 8:00**	**24/06/10 17:00**	**7;14**
30	Pruebas de cobertura	2 days	19/06/10 8:00	20/06/10 17:00	
31	Pruebas escenarios interiores	4 days	20/06/10 8:00	23/06/10 17:00	30
32	Pruebas escenarios exteriores	1 day	24/06/10 8:00	24/06/10 17:00	31

Figura 10.1: Tareas del proyecto

Para no desvirtuar el resumen final, no se han incluido tareas más específicas como pudieran ser la inicialización de los parámetros Bluetooth en cliente y servidor. Con esto se pretende mostrar una visión representativa de lo que ha sido el proyecto y por ello se facilita su visualización.

Para completar esta descripción de la planificación, se muestra el diagrama de Gantt tomando como datos de entrada las tareas anteriormente descritas.

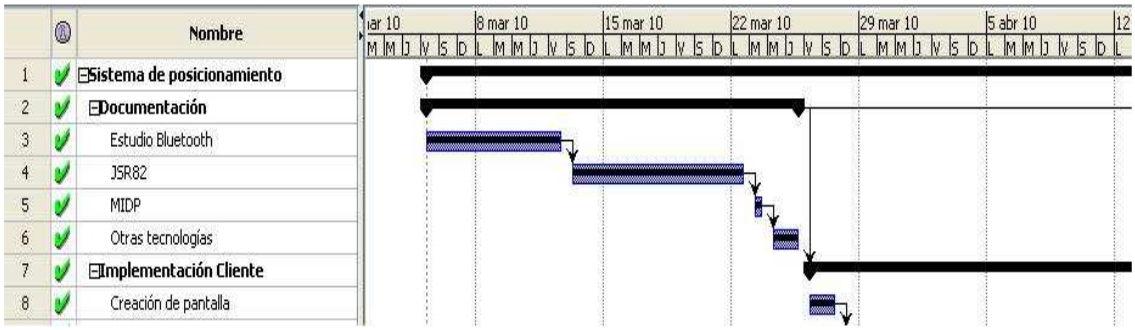

Figura 10.2: Diagrama de Gantt (I)

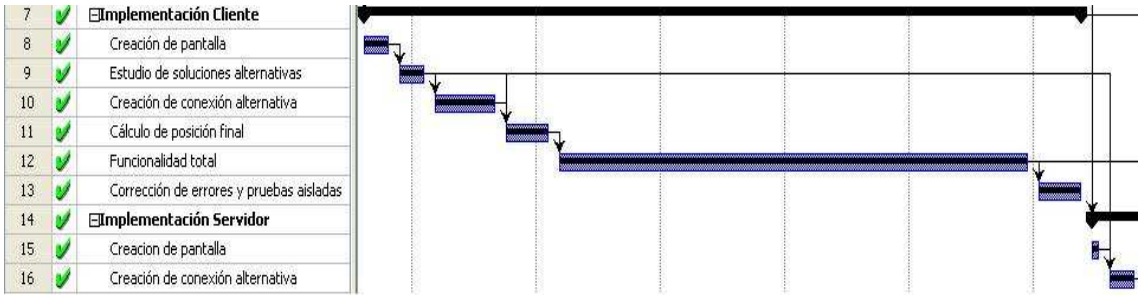

Figura 10.3: Diagrama de Gantt (II)

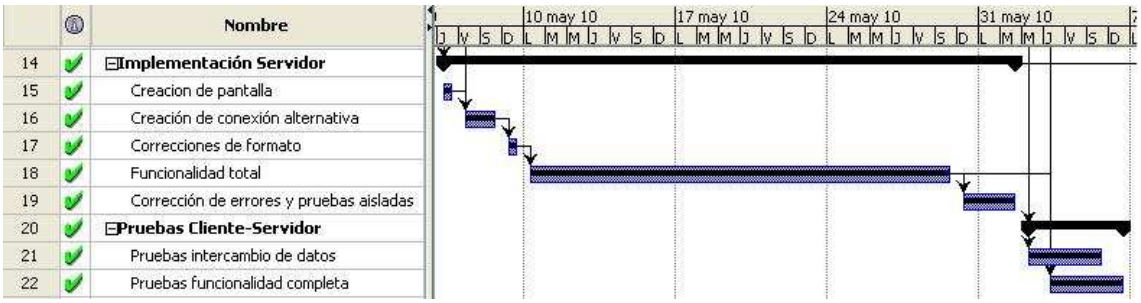

Figura 10.4: Diagrama de Gantt (III)

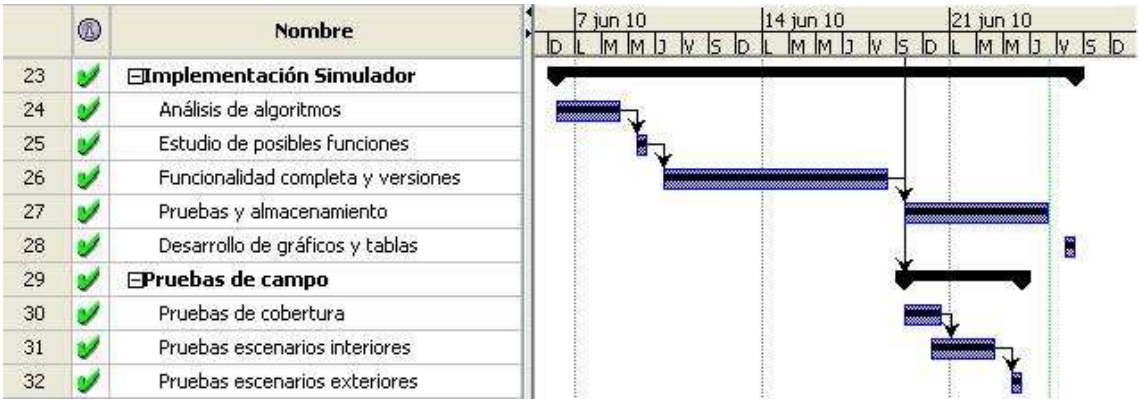

Figura 10.5: Diagrama de Gantt (IV)

Con el fin de aportar una visión global de lo hasta ahora mencionado, incluimos el diagrama de Gantt en su totalidad, sin ningún tipo de cortes.

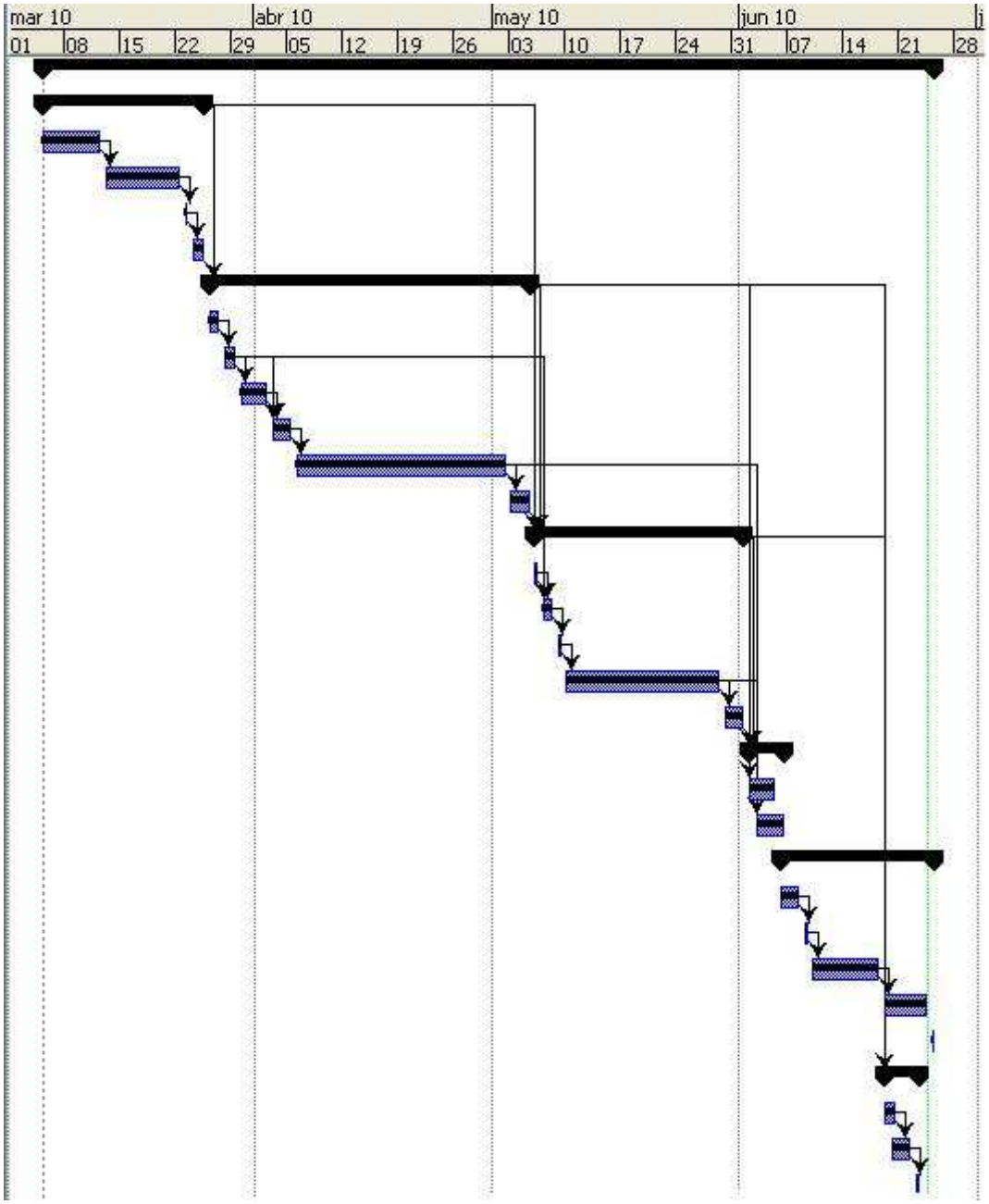

Figura 10.6: Diagrama de Gantt completo

Como se puede observar, varias de las tareas están supeditadas a la realización del cliente y servidor, esto es así ya que se consideran el eje en torno al cual giran las demás tareas del proyecto. Esto es lógico puesto que las pruebas de campo o las de cliente-servidor no pueden ser ejecutadas sin tener terminadas previamente dichas aplicaciones.

Por otra parte, las pruebas reales con terminales móviles se han ido realizando a la par que otras tareas, en este caso de desarrollo de gráficos y tablas. Por este motivo no siguen un orden lineal de tiempos, como puede verse en el diagrama.

Como apunte final al respecto de la planificación cabe remarcar que, pese a los habituales contratiempos sufridos, que en ocasiones han retrasado el ritmo de trabajo, los plazos han sido en mayor o menor medida cumplidos, ya que para facilitar todo el proceso de planificación se añadieron unos estudiados márgenes de error. Por tanto, lo que en un primer momento eran unos plazos estimados, se convirtieron con el devenir del proyecto en, prácticamente, exactos.

Las fechas establecidas son muy cercanas a las reales, ya que ha sido posible contrastar toda la información con un cuaderno de bitácora, constantemente actualizado a la par que el proyecto evolucionaba.

APÉNDICE A Posición De Servidores En Escenarios

Con el fin de poder aportar mayor visibilidad a las pruebas de campo realizadas y definir de un modo más riguroso los escenarios planteados, vamos a incluir las posiciones reales en las que se ubicaron los servidores utilizados en cada caso con relación al punto de referencia definido inicialmente.

Es sencillo intuir que en el escenario planteado para un espacio abierto, las balizas se colocaron en línea recta a lo largo de una calle cualquiera, por lo que sólo fue necesario tener en cuenta una coordenada. Se estudió la distancia óptima entre ellas para que el sistema no perdiera precisión y se lograra una comunicación fluida entre todos los dispositivos. Todas las balizas se colocaron a una altura de unos 170 centímetros.

Baliza	PosiciónX
1	0
2	10
3	20
4	30
5	40
6	50
7	**60**

Tabla A.1: Posición balizas escenario externo

Por otro lado, para tratar los escenarios de interiores, la disposición de las balizas requería de dos coordenadas para la misma planta o tres si se pretendía cambiar de piso. En ese caso la disposición de las balizas en la planta 0 y la 1 se consideró provechoso que fuera la misma. En la planta 2, como se comentó en el capítulo de pruebas de campo, únicamente se colocó una baliza.

La disposición de todas y cada una de ellas se realizó teniendo siempre en cuenta los obstáculos que podían encontrarse, para conseguir de este modo una exigencia máxima del sistema. Se intentó ubicarlas de un modo estudiadamente asimétrico logrando así que no fueran sencillos los cálculos del cliente.

Baliza	PosiciónX	PosiciónY
1	2	3
2	5	3
3	**2**	**4**
4	1	9
5	5	9
6	5	5

Tabla A.2: Posición balizas planta

Baliza	PosiciónX	PosiciónY	PosiciónZ
1	2	4	0
2	8	4	0
3	4	2	0
4	2	4	1
5	8	4	1
6	4	2	1
7	4	2	2

Tabla A.3: Posición balizas entre plantas

Se incluyen también las tres únicas posiciones que se utilizaron para realizar las pruebas interiores en el pasillo de la universidad.

Baliza	PosiciónX	PosiciónY
1	0	2
2	6	2
3	12	2

Tabla A.4: Posición balizas pasillo

APÉNDICE B Bluetooth Assigned Numbers

En este anexo podremos identificar el valor y tamaño de los UUIDs para protocolos y clases de servicio de la librería JSR82. Estos parámetros han sido necesarios para el correcto diseño de la aplicación tanto cliente como servidora.

Nombre	Valor	Tamaño
Base UUID Value	0x1000800000805F9B34FB	128-bit
SDP	0x0001	16-bit
RFCOMM	0x0003	16-bit
OBEX	0x0008	16-bit
HTTP	0x000C	16-bit
L2CAP	0x0100	16-bit
BNEP	0x000F	16-bit
Serial Port	0x1101	16-bit
ServiceDiscoveryServerServiceClassID	0x1000	16-bit
BrowseGroupDescriptorServiceClassID	0x1001	16-bit
PublicBrowseGroup	0x1002	16-bit

OBEX Object Push Profile	0x1105	16-bit
OBEX File Transfer Profile	0x1106	16-bit
Personal Area Networking User	0x1115	16-bit
Network Access Point	0x1116	16-bit
Group Network	0x1117	16-bit
IP	0x0009	16-bit
WSP	0x000E	16-bit
BNEP	0x000F	16-bit
Fax	0x1111	16-bit
UPNP	0x0010	16-bit
TCS-AT	0x0006	16-bit
FTP	0x000A	16-bit

Tabla B.1: Assigned Numbers

APÉNDICE C Manual De Utilización Del Sistema

A lo largo de esta memoria se ha puesto especial interés en mostrar el sencillo funcionamiento del sistema creado, pese a esto para definirlo completamente, a continuación esbozaremos una breve guía de utilización real para el usuario, tanto en el modo cliente como servidor.

En caso de que el usuario quiera ejecutar la aplicación en modo cliente, debe tener claro desde un primer momento el punto de referencia general que se ha tomado para establecer el resto de coordenadas. Es el único dato con el que debe contar para poder comprender la información que le reportará el sistema. Con esto asimilado, el programa cliente le guiará sin problemas sobre cómo debe manejar la aplicación. Búsqueda de dispositivos o cancelación de las mismas aparecen reflejadas claramente en la pantalla del dispositivo. Además, el sistema incorpora varios mensajes de información y error que permiten al usuario conocer el estado del mismo en todo momento. Para dar un mayor contexto a esta guía se muestran las siguientes figuras.

Figura C.1: Inicio cliente

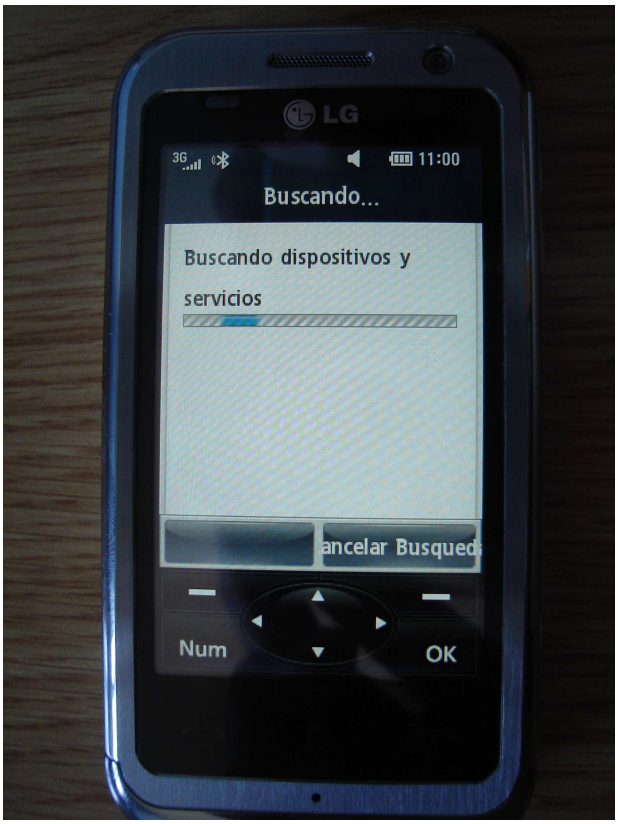

Figura C.2: Cliente buscando

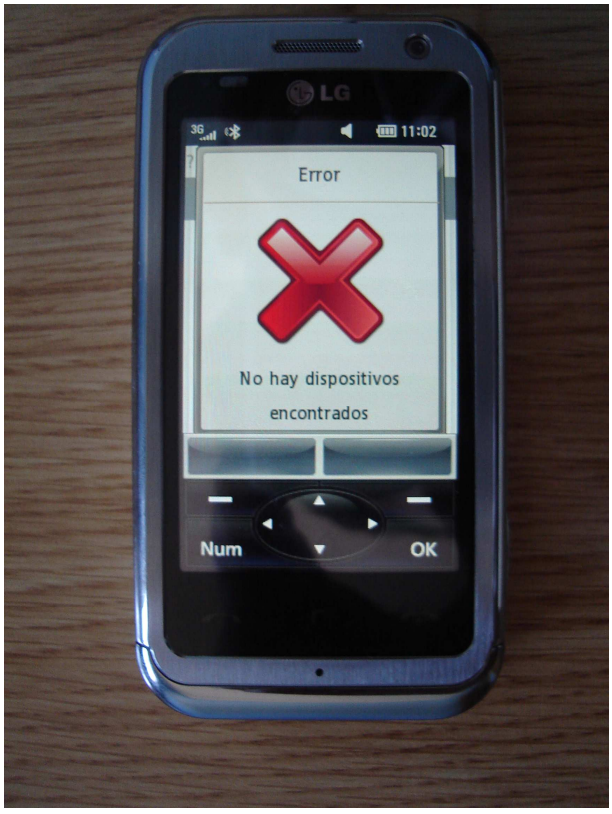

Figura C.3: Cliente no encuentra dispositivos

Por otro lado, si el usuario desea iniciar la aplicación en modo servidor, deberá, igualmente, establecer un punto de referencia a raíz del cual se dispondrán todos los servidores disponibles. Es de vital importancia para el servidor conocer este punto para la correcta ubicación de las balizas, ya que sino fuera así todo el sistema arrastraría este error inicial y por tanto los valores obtenidos no se corresponderían con la realidad.

Otro aspecto a tratar para la correcta configuración del servidor es el formato de la localización. El usuario debe introducir la ubicación de la baliza a colocar con un formato aceptado por el sistema. Se aceptan mayúsculas y minúsculas y se deben introducir las tres coordenadas aunque con alguna no se vaya a trabajar. A su vez, se debe dejar un espacio entre cada coordenada o, en su defecto, una coma.

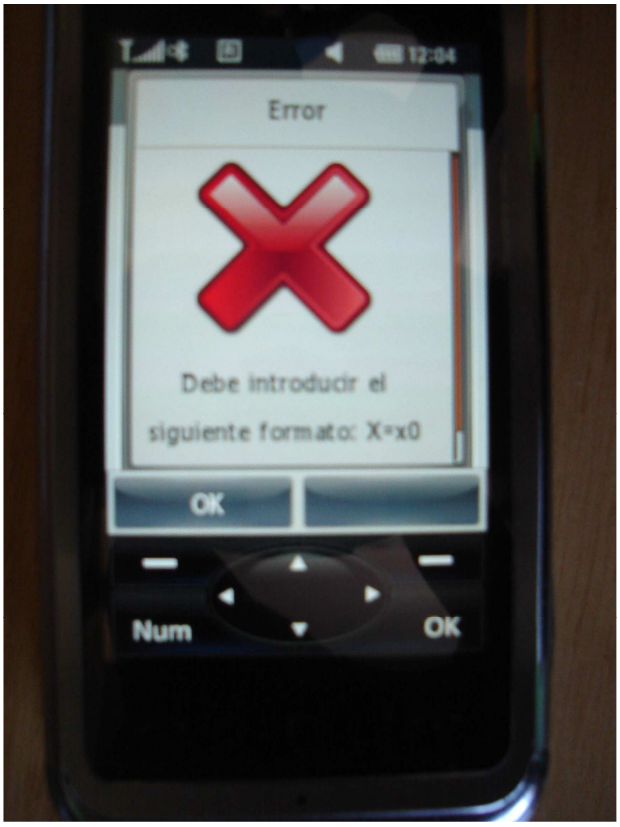

Figura C.4: Error de formato en servidor

Al igual que en el caso del cliente, el sistema mostrará en todo momento mensajes informativos para guiar al usuario en la correcta utilización del sistema. En la figura anterior se puede observar cómo se recomienda al usuario que introduzca el formato correcto para cada una de las coordenadas.

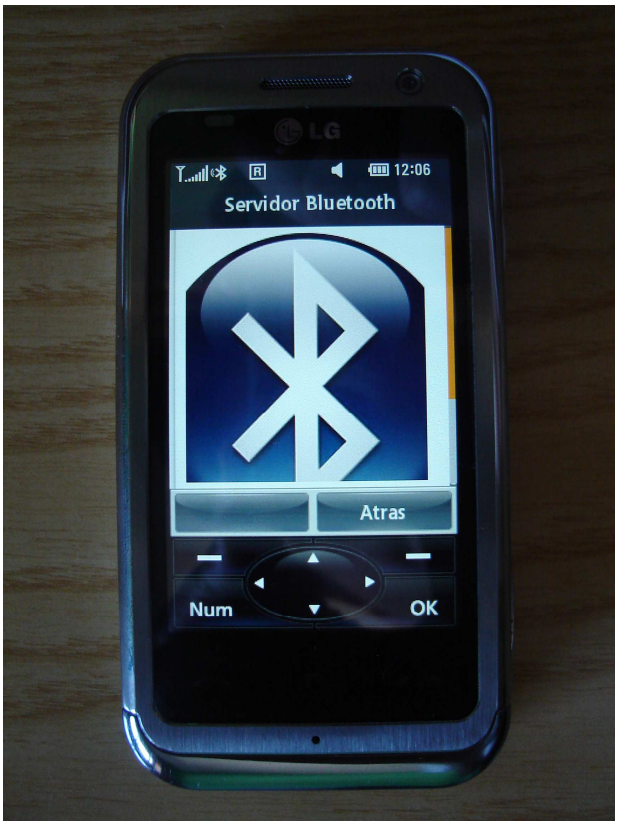

Figura C.5: Servidor con formato correcto ejecutándose

Una vez se ha introducido la información de localización con el formato adecuado, el servidor ya está disponible para enviar dicha información a cualquier cliente que la requiera. En caso de que exista la necesidad de modificar la ubicación del servidor, se deberá pulsar el botón “atrás” para que de nuevo nos aparezca la pantalla de selección.

APÉNDICE D Contenido y Utilización de CD

Adjunto a esta memoria, se entregará un CD con las aplicaciones implementadas para la realización del proyecto. Como se comentó en capítulos anteriores, el proyecto consta de un sistema de posicionamiento para teléfonos móviles, que puede actuar como servidor o cliente, junto con una aplicación simuladora de errores de precisión del mismo que puede presentar varias versiones. Además, todas las pruebas realizadas, tanto en simulación como en escenarios reales, serán igualmente incluidas.

Aplicación Bluetooth

Al tratarse de la aplicación principal del proyecto, se van a describir detalladamente los pasos necesarios para conseguir su instalación y utilización, así como los entornos y herramientas requeridos.

Descarga e instalación del entorno

Para que la aplicación pueda correr sin problemas en nuestro ordenador deberemos, primeramente, descargar e instalar de manera correcta el entorno de desarrollo correspondiente.

A su vez, previo a la instalación de entornos específicos, es necesario tener instalado en nuestro PC una versión del estándar de Java (J2SE). Ya que vamos a trabajar con versiones actualizadas, es recomendable descargarse la última versión disponible de la página http://java.sun.com/javase/downloads/index.jsp, en este caso la denominada JDK 6, actualización 20. En nuestro caso contamos con la actualización 12. En este paquete podremos encontrar la versión 1.6 del JDK. Versiones más antiguas no podrán funcionar con el paquete de J2ME descargado.

En el caso que nos ocupa, al tratarse de una aplicación creada para ejecutarse en terminales móviles deberemos contar con J2ME (Micro Edition). A su vez, necesitamos un simulador de terminales con el que podamos comprobar el funcionamiento del sistema. Ambos requisitos los podemos encontrar en un mismo paquete de descarga. Para ello debemos entrar en la página de descargas de Java (http://java.sun.com/javame/downloads/index.jsp) y elegir uno de los kits de desarrollo de software. En esta página podemos contrar con el conocido *Wireless Toolkit* o con un nuevo paquete llamado *Java Platform Micro Edition SDK*, versión 3.0. Este último es más recomendable ya que se trata de una evolución del anterior e incorpora más funcionalidades, habiéndose eliminado los bugs iniciales. Ambos están disponibles para XP, Vista o Mac.

Conviene incluir algunos de los requisitos que podrían ser limitantes, dado que en cuanto a sistema son algo “altos”.

1Ghz de CPU
1GB de RAM
JDK 1.6 o superior.

El proceso de instalación es sencillo ya que los pasos están bien definidos y explicados. La única complicación posible estriba en elegir la carpeta en la que hemos instalado previamente el JDK. Esta selección debe ser correcta para el adecuado funcionamiento del sistema. Normalmente la podemos encontrar en la carpeta raíz o en archivos de programa, Java y con nombre *jdk1.6.0_versión*.

En el momento en el que el paquete quede instalado el usuario podrá probar y simular el sistema de cualquier manera posible.

Utilización

Para ello se debe importar el proyecto que aparece en el CD o insertarlo en la carpeta de proyectos de la plataforma (.\Mis Documentos**JavaMESDKProjects**). Esta última opción es preferible por su simplicidad. Por tanto los pasos serán los siguientes:

Colocar la carpeta LocalizacionBT en la carpeta de proyectos de la plataforma.
Abrir el entorno de desarrollo instalado.
Pulsar el botón O*pen Project* situado en la parte superior izquierda de la pantalla.
Seleccionar el proyecto anteriormente insertado y abrirlo.

Figura D.1: Pulsar botón open

Una vez que nos encontramos con la aplicación y sus diferentes clases podremos modificarla, guardarla, compilarla o ejecutarla. Un atajo para la ejecución es el botón verde colocado en la parte superior izquierda, en el panel de herramientas. Este botón nos permitirá ejecutar rápidamente la aplicación y, a continuación, aparecerá como interfaz de usuario un terminal móvil que ejecuta nuestro programa. Para poder compilar existe otro botón de atajo en forma de martillo.

Figura D.2: Botones de barra de herramientas

Para que estos botones tengan utilidad debemos definir nuestro proyecto como principal. La disposición de la pantalla muestra en la zona lateral izquierda superior un cuadro con los proyectos actuales. Pinchamos con el botón derecho sobre nuestro proyecto y seleccionamos la opción *set as main Project.*

Para trabajar con el proyecto sin hacerlo principal es necesario compilar y ejecutar desde las pestañas superiores, en concreto la pestaña Run.

Una vez ejecutada la aplicación, para insertar la localización en el servidor es necesario pulsar el botón menú de la parte inferior derecha del terminal. La opción que debemos elegir es la segunda, *insertar localización.*

Como es necesario el uso de, al menos, tres dispositivos para probar el sistema (un cliente y dos servidores) y el entorno no permite ejecutar tres terminales del mismo tipo, es necesario ejecutar los restantes dispositivos bajo otro tipo de terminales.

En la zona lateral izquierda, en el cuadrado situado debajo de los proyectos aparecen todos los tipos de terminales. Cada uno de estos tiene unas características particulares, por tanto no todos soportarán la aplicación. Los tres terminales que son capaces de soportarla son *DefaultCldcPhone1*, *DefaultCldcPhone2* y *ClamShellCldcPhone1*, todos incluidos en el grupo CLDC.

La versión MIDP utilizada es la 2.1 y la CLDC es la 1.1, es decir, las versiones más actuales posibles en cada caso.

Para consultas más avanzadas, el programa cuenta con un muy buen archivo de ayuda que se podrá consultar en cualquier momento en la pestaña Help.

Como apunte final, la imagen D.2 incluye la apariencia total del entorno de ejecución.

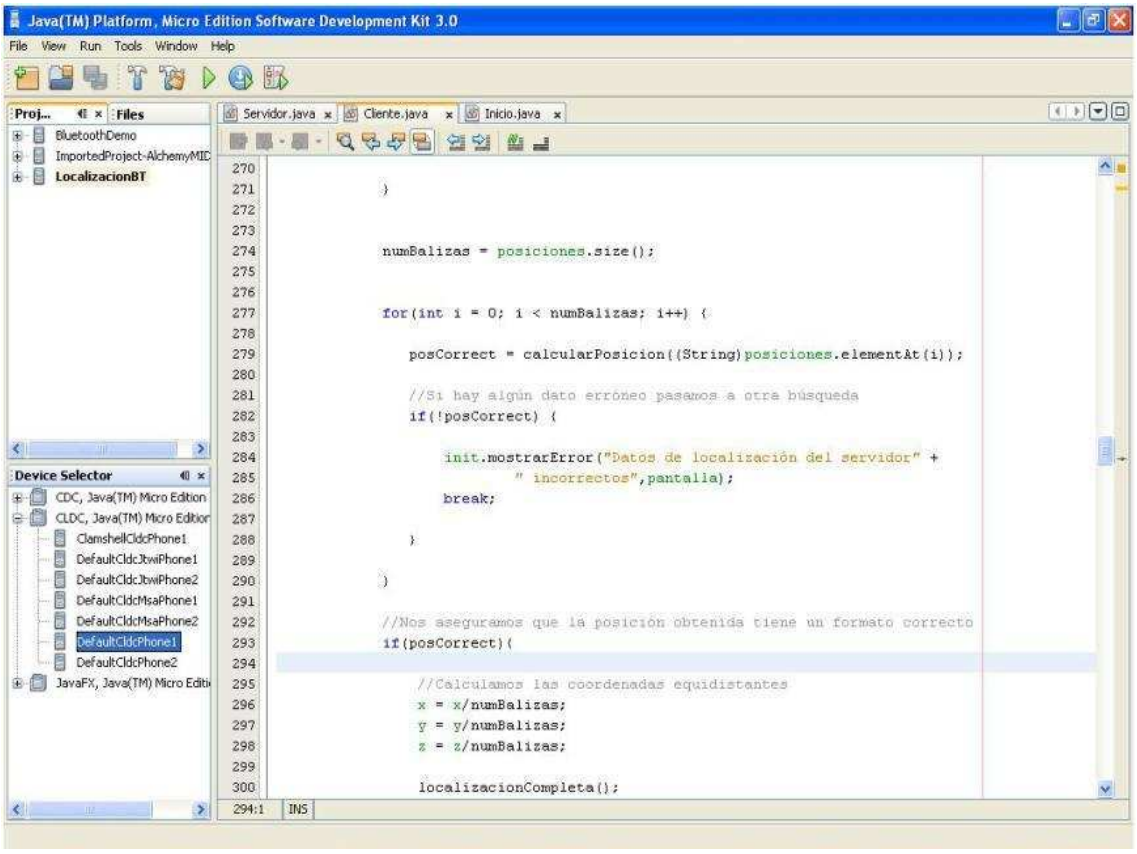

Figura D.3: Vista completa de entorno de ejecución

Aplicación simuladora

Esta aplicación adicional recoge los errores de precisión. Al tratarse de un programa Java de ejecución normal, solamente es necesario tener instalado un JDK de cualquier versión. Dado que anteriormente se instaló la versión 1.6 para el entorno J2ME, se ha trabajado con ésta.

Descarga e instalación del entorno

Para ejecutar la aplicación se procedería del modo habitual por línea de comandos. Sin embargo, para ofrecer unas mayores prestaciones y en caso de que exista la necesidad de realizar alguna modificación, se recomienda la utilización de Eclipse ya que no necesita instalación. Se puede descargar de http://www.eclipse.org/downloads/, siendo la última versión la 3.6.

Utilización

Los pasos a seguir para poder utilizar la aplicación simuladora con Eclipse son los mismos que en el caso anterior, ya que se necesita crear un proyecto.

Como anteriormente, debemos colocar la carpeta proporcionada en el CD (PruebaAlgoritmos) en la carpeta de proyectos de Eclipse. En este caso tiene el siguiente camino: .\Documents and Settings\Usuario**workspace**.

El uso de este entorno de desarrollo es algo parecido al aportado por Java en lo que a botones de atajo se refiere, si bien, añade una infinidad de funcionalidades adicionales. Para compilar nos encontramos con un botón en forma de insecto y para ejecutar tenemos el botón verde de su derecha. Además podemos contar con puntos de ruptura y varias vistas, siendo muy útil la vista en modo depuración, ya que podemos ver en todo momento el valor de todas las variables creadas en la aplicación o parar y volver a reanudar la ejecución.

Únicamente mostraremos una visión general de la interfaz de usuario que ofrece este potente entorno de desarrollo tanto para vista Java como para vista depurador. Con ello se pretende familiarizar al usuario con su funcionamiento.

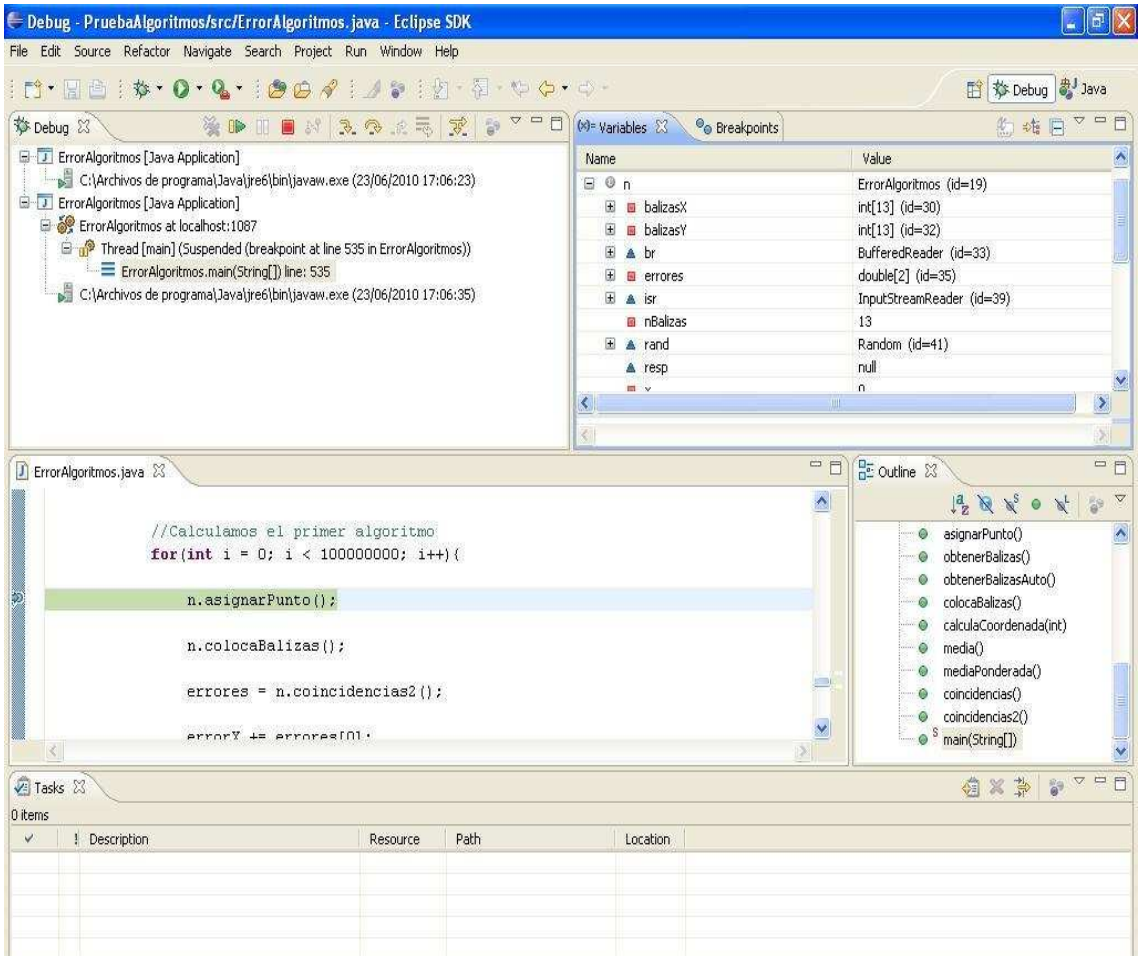

Figura D.4: Vista depurador Eclipse

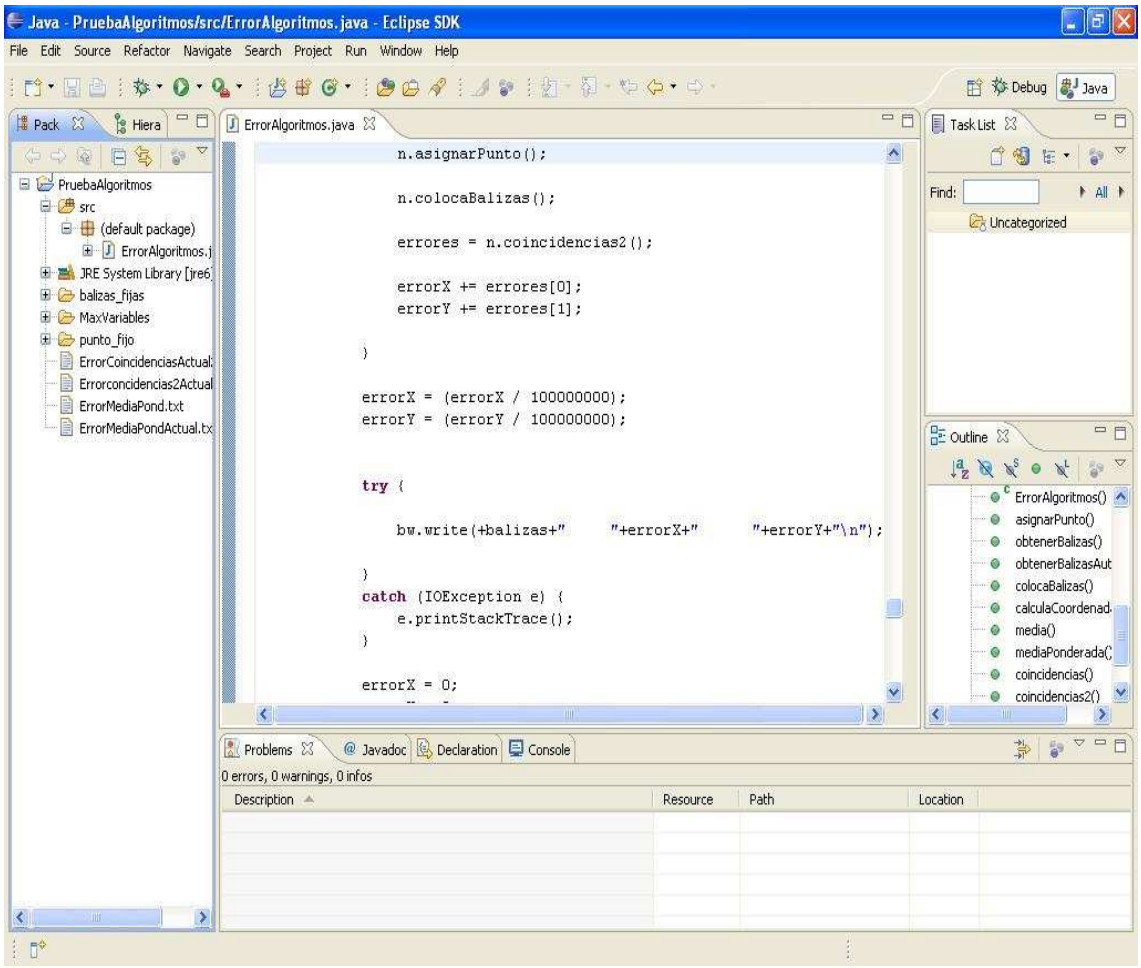

Figura D.5: Vista Java Eclipse

En caso de ejecutar la versión automática de la aplicación simuladora, no es necesario realizar ninguna otra acción, los archivos creados se podrán encontrar en el workspace de Eclipse. Si por el contrario el usuario desea introducir el número de balizas, deberemos dar este paso previo al almacenamiento de los datos reportados.

Para comprobar los diferentes algoritmos implementados, se debe cambiar en el método `main` el nombre del algoritmo que se pretenda invocar (`media`(), `mediaPonderada`(), `coincidencias`() o `coincidencias2`()). A su vez, se necesitará cambiar el nombre del archivo dónde se almacenan los datos reportados. Si se desean incorporar variaciones tales como dejar las balizas fijas o la posición del usuario, únicamente habrá que sacar el método correspondiente del bucle de cálculo (`asignarPunto`(), `colocaBalizas`()).

En la carpeta Datos_obtenidos, incluida en el CD dentro del proyecto de simulación, vienen contenidos todos los datos reportados por la aplicación en sus diferentes versiones y algoritmos. Es decir, todas las pruebas de simulación válidas realizadas. Además, en la carpeta Pruebas se encuentran estos datos tratados con sus correspondientes tablas y gráficas, así como los valores obtenidos en las pruebas reales y de cobertura.

GLOSARIO

WiFi: *Wireless Fidelity*, tecnología inalámbrica de área local, estándar 802.11.

IrDA: *Infrared data asociation*, tecnología inalámbrica de transmisión de datos mediante infrarrojos.

ZigBee: tecnología inalámbrica para comunicación de bajo consumo basada en el estándar 802.15.4, redes de área personal.

JDK: *Java Development Kit*, paquete de desarrollo de Java.

J2ME: *Java Micro Edition*, especificación de Java para desarrollo de software para dispositivos móviles.

J2SE: *Java Standard Edition*, especificación para el desarrollo de cualquier tipo de aplicación Java.

JSR82: *Java Specification Request*, especificación de Java para desarrollo de aplicaciones Bluetooth para dispositivos móviles.

JSR179: especificación de Java para desarrollo de aplicaciones de localización.

MIDP: *Movile Information Device Profile*, especificación Java para el desarrollo de aplicaciones en dispositivos móviles.

MIDlet: Aplicación para dispositivos móviles implementada mediante MIDP.

A-GPS: *Assisted glotal Positioning System*, sistema de GPS asistido para dispositivos móviles.

GPS: *Global Positioning System*, sistema de posicionamiento global.

ETSI: *European Telecommunications Standards Institute*, organización de estandarización para las telecomunicaciones.

HTTP: *Hypertext Transfer Protocol*, protocolo usado para las transferencias en la red de Internet.

URL: *Uniform Resource Locator,* caracteres usados para nombrar recursos en Internet.

RS-232: *Recommended Estándar 232,* interfaz para comunicaciones en puerto serie.

SR: *Service Record,* registro para almacenar los parámetros de un servicio Bluetooth.

DE: *Data Element,* registros que contienen los atributos de un servicio Bluetooth. Se definen mediante pares identificador, valor.

BT: *Bluetooth,* tecnología inalámbrica de comunicación a corta distancia.

SPP: *Serial Port Profile,* perfil Bluetooth que emula una comunicación RS-232 de puerto serie.

SDP: *Service Discovery Protocol,* proporciona a las aplicaciones Bluetooth los servicios disponibles y sus características.

BCC: *Bluetooth Control Center,* controlador que define varios valores de la pila Bluetooth.

SDDB: *Service Discovery Database,* base de datos que almacena los Service Record.

UUID: *universally unique identifiers,* identificadores únicos de protocolos y servicios Bluetooth.

GIAC: *General InquiryAccess Code,* modo de búsqueda general para cualquier dispositivo Bluetooth.

EDR: *Enhanced Data Rate,* técnica incorporada en la versión 2.0 de Bluetooth que permite mejorar las velocidades de esta tecnología.

A2DP: *Advanced Audio Distribution Profile,* perfil Bluetooth que define cómo propagar Streams.

SIG: *Special Interest Group,* grupo privado formado por una gran variedad de compañias que controla todo lo relacionado con la tecnología Bluetooth.

COITT: *Colegio Oficial de Ingenieros Técnicos de Telecomunicaciones.*

BIBLIOGRAFÍA

1 Bluetooth versión 2.1, última consulta 25/5/10.
http://www.tgdaily.com/networking-features/33221-bluetooth-spec-updated-to-version-21edr

2 Artículo del SIG sobre las últimas actualizaciones Bluetooh, última consulta 15/3/10.
http://www.bluetooth.com/English/Press/Pages/PressReleasesDetail.aspx?ID=4

3 Página sobre las especificaciones básicas de Bluetooth, última consulta 20/3/10.
http://www.bluetooth.com/English/Technology/Pages/Basics.aspx

4 Página oficial del transductor Bluetooth LMX5252, última consulta 10/3/10.
www.national.com/appinfo/wireless/files/LMX5252_prodbrief.pdf

5 Especificaciones Bluetooth y pila de protocolos, última consulta 3/6/10.
http://www.palowireless.com/infotooth/tutorial.asp

6 Estándar del ETSI, última consulta 12/5/10.
ETSI, TS 101 369 (GSM 07.10) version 6.3.0

7 Página de desarrollo de aplicaciones Bluetooth en Java, última consulta 29/5/10.
http://developers.sun.com/mobility/midp/articles/bluetooth2/

8 Especifiación Bluetooth del perfil SPP, última consulta 9/5/10.
SPP_SPEC_V12

9 Documento que define el API de Bluetooth para Java, 29/5/10.
JSR82-spec_1.1.1 (bluetooth-1.1.1-mrel2-javadoc)

10 Página de Java para la utilización de JSR82, última consulta 15/6/10.
http://developers.sun.com/mobility/apis/articles/bluetoothcore/index.html

11 Página oficial del colegio de ingenieros técnicos de telecomunicaciones, última consulta 25/6/10.
http://www.coitt.es/index.php?page=noticias_coitt_reg&icod=181

12 Bradford W. Parkinson, **Global Positioning System: Theory and Applications**, 1996.

13 Timothy J. Thompson, C Bala Kumar, Paul J. Kline, **Bluetooth application programming with the Java APIs Essentials Edition**, 2008.

14 Api de Java para JSR82, última consulta 28/6/10 bluetooth-1.1.1-mrel2-javadoc/allclasses-frame-bt.html

15 Virginia Téllez García Moreno, **Análisis de las prestaciones de la tecnología Bluetooth**, 2004.

16 Pedro Daniel Borchez Juzgado, **Java 2 Micro Edition Soporte Bluetooth, versión 1.0**, 2004.

El que no considera lo que tiene
como la riqueza más grande,
es desdichado, aunque sea
dueño del mundo.

Epicuro, filósofo Griego.